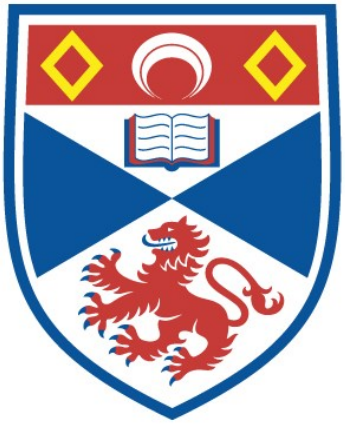
APPLYING NAMED DATA NETWORKING IN MOBILE AD HOC NETWORKS

Percy Dante Perez Aruni

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2019

Applying Named Data Networking in Mobile Ad Hoc Networks.

Percy Dante Perez Aruni

University of
St Andrews

This thesis is submitted in partial fulfilment for the degree of

Doctor of Philosophy

at the University of St Andrews

March 2019

Abstract

This thesis presents the Name-based Mobile Ad-hoc Network (nMANET) approach to content distribution that ensure and enables responsible research on applying named data networking protocol in mobile ad-hoc networks. The test framework of the nMANET approach allows reproducibility of experiments and validation of expected results based on analysis of experimental data. The area of application for nMANETs is the distribution of humanitarian information in emergency scenarios. Named-Data Networking (NDN) and ad-hoc mobile communication allow exchange of emergency information in situations where central services such as cellular towers and electric systems are disrupted.

The implemented prototype enables researchers to reproduce experiments on content distribution that consider constraints on mobile resources, such as the remaining power of mobile devices and available network bandwidth. The nMANET framework validates a set of experiments by measuring network traffic and energy consumption from both real mobile devices and those in a simulated environment. Additionally, this thesis presents results from experiments in which the nMANET forwarding strategies and traditional wireless services, such as hotpost, are analysed and compared. This experimental data represents the evidence that supports and validates the methodology presented in this thesis.

The design and implementation of an nMANET prototype, the Java NDN Forwarder Daemon (JNFD) is presented as a testing framework, which follows the principles of continuous integration, continuous testing and continuous deployment. This testing framework is used to validate JNFD and IP-based technologies, such as HTTP in a MANET using the OLSR routing protocol, as well as traditional wireless infrastructure mode wireless.

The set of experiments executed, in a small network of Android smart-phones connected in ad-hoc mode and in a virtual ad-hoc network simulator show the advantages of reproducibility using nMANET features. JNFD is open source, all experiments are scripted, they are repeatable and scalable. Additionally, JNFD utilises real GPS traces to simulate mobility of nodes during experiments. This thesis provides experimental evidence to show that nMANET allows reproducibility and validation of a wide range of future experiments applying NDN on MANETs.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Alexander Voss for his immense support, help, and specially for his time and patience during all this time. Thank you so much. At the same, I would like to thanks to Dr. Marzieh Asgari-Targhi and little Sophie Voss for their moral support during hard times of my PhD.

To Dr. Ishbel Duncan for her support and encourage as my second supervisor. To Alex Bain and Jacky Lawson for their administrative support at the University. I would like to thank to many people I met and shared during this period. To Jim Park, Nicol Thomson, David Letham and Stuart Norcross for all the technical support in this thesis. To Juanjo Mendoza, Daniel Rough, Long Thai for the code review of the initial version of my prototype. To Dr. Vinodh Rajan, Judith Malcolm, and Andrea Rosales for the amazing company in the school. To Amjad Al Tobi, Haifa al Nasseri for their feedback in the research discussions. To Ildiko Pete for proof read the initial draft chapters and to Daniel Rough for proof reading the final corrections. To Issac and Ann in supporting me in the most hardest times. To Chris Schwab, Rebeca and Esme for all their constant encourage and long chats about life. To Juliet, Victor and Tim for their advices and time during hard moments. To Prof Harry and Rosalind for let me feel like home here in St Andrews. To Marcela, Margareth and Priscila for the motivation I received before and during my PhD.

I may missed some of the names, I just wanted to say thank you for all small and big details, but specially for the life time you all spent on me in this life. Finally, I would like to remark my gratitude to my sister Eliana, my brother Roy, my nephew Mathias, specially to my Mum Demetria and my Dad Mario for all the incredible support during my absence at home.

Funding acknowledgement

I extend my most respectful gratitude to the funders of this project, and all people who make this possible. This work was supported by the University of St Andrews, the School of Computer Science.

Thank you so much. Yuspagara!

Declaration

Candidate's Declarations

I, Percy Dante Perez Aruni, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 54,430 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree.

I was admitted as a research student at the University of St Andrews in April 2013.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date: March 27th, 2019

Signature of candidate:

Supervisor's Declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date: March 27th, 2019

Signature of supervisor:

Permission for Electronic Publication

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis. I, Percy Dante Perez Aruni, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Date: March 27th, 2019

Signature of candidate:

Signature of supervisor:

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Percy Dante Perez Aruni, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Date: March 27th, 2019

Signature of candidate:

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force.

We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below.

The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs:

No embargo on underpinning research data or digital outputs.

Date: March 27th, 2019

Signature of candidate:

Signature of supervisor:

Dedicate to my inspirations in life
my little nephew Mathias,
my little "chato" Roy,
my little "ratona" Eliana,
but specially this is dedicated
to my
Dad Mario Hermogenes Perez Limachi
and my
Mum Demetria Inocencia Aruni Rojas de Perez.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AODV Ad-hoc On Demand Distance Vector

AOSP Android Open Source Platform

BATMAN Better Approach To Mobile Ad-hoc Networking

CCN Content Centric Networking

CDN Content Distribution Network

CS Content Store

DONA United Nations

FEMA United States Federal Emergency Management Agency

FIB Forwarding Interest Base

ICN Information Centric Networking

ILNP Identifier Locator Network Protocol

ITU International Telecommunication Union

JNFD Java NDN Forwarder Daemon

JNI Java Native Interfaces

LFBL Listen First Broadcast Later protocol

LSA Link State Advertisements

mA Milliamperes

MAC Medium Access Control

mAh Milliamperes-hour

MANETs Mobile Ad-hoc Networks

NDN Named Data Networking

NFD NDN Forwarding Daemon

NLSR Named-data Link State Routing Protocol

nMANET NDN for MANETs

OCHA United Nations Office for the Coordination of Humanitarian Affairs

OLSR Optimized Link State Routing

PIT Pending Interest Table

PKI Public Key Infrastructure

SMS Short Message Service

SPAN Smart Phone Ad-hoc Network

TCP Transmission Control Protocol

UDP User Datagram Protocol

UN United Nations

URL Uniform Resource Locator

ZRP Zone Routing Protocol

CHAPTER ONE

INTRODUCTION

1.1 Problem statement

A common scenario described by the reports of international humanitarian organisation such as the United Nations involves the high dependency on centralised communication infrastructures. Victims and responders are connected with each other through intermediate systems such as cellular towers from service providers, public Wi-Fi routers or other wired or wireless platforms. However, it is not possible to guarantee the availability of these intermediate infrastructures and the services they provide during an emergency scenario. Cases such as the 2010 Haiti and 2011 Tohoku earthquake are evidence that electric power and access to Internet in the disaster area were often cut, making services unavailable and remaining communication modules useless for victims and responders. Consequently, humanitarian relief organisations highlight the importance of empowering victims and responders in emergency scenarios with more people oriented communication platforms, rather than a high dependency on traditional centralised infrastructures such as cellular networks [Crowley and Chan, 2011].

One of the most dynamic technologies reported by the International Telecommunication Union (ITU) is mobile communications. According to the ITU, by 2015 the overall penetration of mobile broadband subscriptions have increased more than twelve times since 2007 and mobile devices are gradually becoming part of daily life in many societies across the world. The trends of this technology make mobile devices an attractive technology candidate to be applied in emergency scenarios. This is especially the case as mobile devices are often equipped with communication channels that can operate independently of infrastructure in Mobile Ad Hoc Networks (MANETs).

Even though an extensive literature exists about MANETs, implementations still have limitations. Mobility of nodes causes connections to be dropped and overheads of routing protocols drain

battery power, which is particularly important in emergency situations. The focus of this thesis is to present an approach that enables responsible research to address the limitations of MANETs for data distribution.

A simplified version of the problem is illustrated by Figure 1.1, which depicts the scenario where a victim requires emergency information from a responder. The victim sends a request through one or more than one intermediate victim's mobile phone and the information is retrieved by utilising these intermediate mobile phones. The problem consists in how the requester can efficiently retrieve the information from the responder through intermediate mobile nodes and minimizing the use of mobile resources, such as energy from batteries. In addition to the limited resources of mobile devices, another challenge includes the unpredictable mobility of these intermediate mobile devices in humanitarian scenarios.

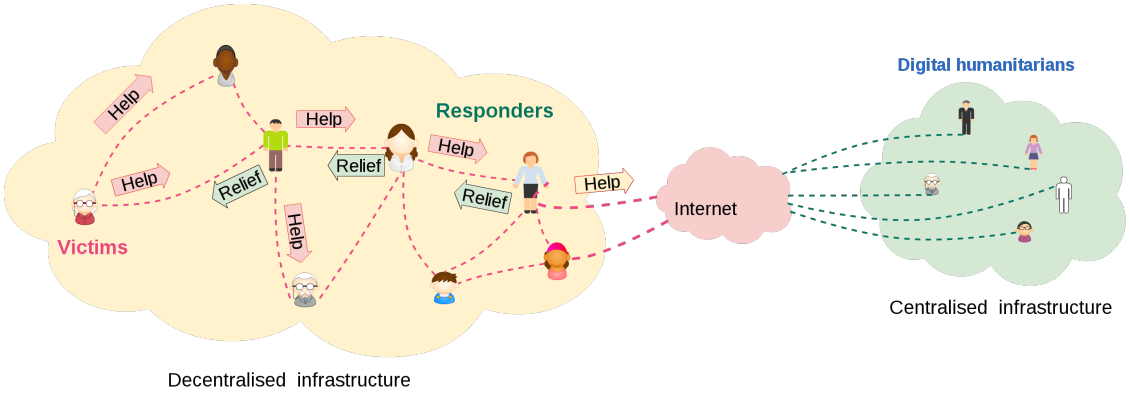


Figure 1.1: Data distribution problem.

1.2 Motivation

The emergency cases discussed in Chapter 2 emphasise the importance of mobile communication and the need to create people-based approaches that allow, in particular, victims to act autonomously. Victims in the emergency area play a critical role: victims have a better understanding of the affected area, they are aware of the nearby available resources, they can communicate using the local language, and what is more, they have developed social relationships prior to the emergency. Therefore, by empowering them with ad hoc local communication, victims can act independently and contribute more accurate relief information in combination with responders and digital humanitarians [Meier, 2015].

Mobile ad hoc communication has the potential to have a significant impact in emergency scenarios, and it aims to provide decentralised communication between mobile devices connected in ad hoc mode. Current solutions [Murthy and Manoj, 2004, Boukerche et al., 2011] are mainly

based on utilising the Internet Protocol (IP) stack, where the addresses of mobile devices (nodes) from the origin and the destination are required to establish communication and exchange of information.

In general, IP-based approaches have limitations that make the creation of effective and efficient MANETs a challenging task. The lack of a caching infrastructure means that content is always retrieved from its originating node, causing duplication of traffic when multiple users access the same content. The origin node replies with the same information for each user. A typical example is when a set of mobile devices request the same video from Youtube or request the same image tiles from a mapping service.

Another limitation that keeps MANET from working effectively is the difficulty of IP address assignment as nodes in a MANET can leave and join the network at any time. IP-based MANETs require a mechanism to assign unique IP addresses to joiners and to make available to the network the IP addresses of nodes that are leaving the network.

To successfully retrieve information from the destination node, IP-based approaches require high availability of the route between the origin and the information provider. In the case that one of the links between end nodes is broken, further requests cannot be satisfied until the link is fixed or an alternative path is discovered. As nodes in MANETs have high mobility, a guarantee of stable links between nodes is difficult to achieve as the topology of the network constantly changes.

A routing protocol is required to find paths between two end nodes, and to provide a mechanism to update routing information between them. As the mobility of nodes produces changes to the topology of the network, routing information has to be either constantly updated or established when required, causing overheads or delays.

Additionally, routing protocols in MANETs affect resource management as they need to make efficient use of remaining energy from batteries and need to work with limited amounts of memory. The network lifetime is highly impacted by the methodology utilised to manage power from batteries. These challenges keep MANETs from being deployed in real scenarios such as emergencies.

This thesis introduces Information Centric Networking (ICN)[1] [Jacobson et al., 2009a, Trossen et al., 2010, Tyson et al., 2012] as a starting point to create an infrastructure-less communication network before, during and after an emergency or natural disaster. ICN is an alternative approach to networking that contrasts with the traditional connection-oriented

[1] https://irtf.org/icnrg

networking model used by the TCP/IP stack. Instead of addressing nodes in a network, ICN technologies address data objects, allowing them to be retrieved from one of potentially many nodes in the network. Content distribution and caching therefore become essential parts of the network architecture instead of being application-level add-ons. The Named Data Networking[2] (NDN) protocol [Zhang et al., 2014] is a concrete implementation of an ICN that I am using in this thesis.

The case for applying ICN instances, such as NDN, in emergency scenarios is described in a draft document on "Research Directions for Using ICN in Disaster Scenarios"[3], published by an IETF working group. It outlines the benefits of ICN features, such as caching, name-based forwarding, and security. These are further discussed in Chapter 3.

To the best of my knowledge and at the time of writing, there is no extensive experimental literature on the use of NDN in MANETs, in particular for energy-efficient approaches. Therefore, the motivation of this thesis is to contribute to the NDN community by providing experimental evidence regarding how to ensure reproducibility of data distribution through NDN in MANETs.

1.3 Scope

The scope of this thesis is to introduce an implementation of NDN for MANETs and provide experimental evidence on the effectiveness and reproducibility of data distribution and on energy consumption. The thesis outlines general principles of using NDN in MANETs, suggesting a category of systems I call *nMANETs*.

Data distribution in NDN networks is understood in terms of a consumer-producer relationship with intermediate nodes serving as network caches that may send data to the consumer directly without involving the producer if they have the data requested. A consumer sends an *interest* to nearby nodes in order to retrieve the desired content. These nodes in turn forward the interest to other nodes around them using a *forwarding strategy* until a node is reached that can fulfill the request. The *interest* may or may not reach a producer.

This thesis presents *nMANET forwarding strategies* and evaluates them by running a prototype implementation of an nMANET forwarder on Android smartphones and on virtual networks through the Mininet-WiFi [Fontes et al., 2015] network simulator. The metrics collected to quantify differences include completeness, correctness, responsiveness, network traffic and energy consumption of batteries.

[2] http://named-data.net/

[3] https://tools.ietf.org/html/draft-irtf-icnrg-disaster-02

Even though NDN aims to eliminate the use of IP addresses, the nMANET experimental prototype is developed to be functional over IP but this is used only for local hop communication. Deploying NDN over IP facilitates the gradual integration into today's IP-based infrastructures and using IP-based tooling. In the future, NDN protocols can be implemented directly on top of data link layer technologies, replacing IP.

This thesis does not aim to cover extensively the security aspects of NDN in MANETs. Security aspects in MANETs and in NDN would be topics suitable for a PhD dissertation in its own right and so cannot possibly be covered as part of this one. However, they are mentioned where necessary and are discussed as future work.

1.4 Research questions and challenges

The principal challenge is to ensure and enable responsible research on applying named data networking in MANETs through reproducibility. Therefore, researchers can test and experiment NDN approaches to solve MANETs limitations.

The main research questions are as follows:

1. Are there implementations of applying Information Centric Networking in MANETs.

2. Are these implementations ensure testing and reproducibility for data distribution in MANETs?.

3. How to validate a new implementation of NDN in MANETS?

1.5 Novel contributions

The thesis makes the following novel contributions:

Firstly, it provides an empirical evidence on how to ensure and enable responsible research on data retrieval in a MANET using NDN protocol. This thesis provides scenarios that can be retested and reproduced. Therefore, it represents a starting point to further elaborate and refine the basic approach designed and implemented in here. In these scenarios, Name-based networking is shown to consume fewer network resources compared with traditional IP-based MANET implementations.

Secondly, to the best of my knowledge, the prototype of nMANET is the first practical implementation of named-data networking in MANETs, available for Android smartphones as well as in Linux-based environments.

Thirdly, the approach to testing and experimenting can be reused for testing alternative implementations in virtual networks such as those provided by Mininet[4] [Lantz et al., 2010] and Mininet-WiFi[5] [Fontes et al., 2015]. In addition to the documentation generated from experiments and reverse engineering, nMANET offers to researchers and developers a methodology to evaluate resource-based forwarding strategies that can be utilised to create new forwarding strategies for nMANETs. The source code of the nMANET forwarder implementation is open source and free to use. The data sets collected from experiments and the technical procedure to reproduce experiments are available as well.

nMANET also provides a virtual environment to scale up and reproduce tests and experiments through what is called Mini-JNFD. Mini-JNFD is based on Mininet-WiFi and allows continuous delivery through a tool, the Testing and Experiment Manager, that creates the virtual environment, compiles recent source code modifications, executes the experiment and collects the results in an aggregated summary. Through this, nMANET contributes to improve on the low percentage of repeatable publications reported by Kurkowski et al. [Kurkowski et al., 2005].

The thesis provides additional empirical information about NDN packets and packet exchanges that go beyond the existing NDN documentation. This information is the conclusion of the analysis of the experimental data collected between NDN forwarders, producers and consumers. The collected packets were analysed octet by octet and decoded using the available description in the NDN technical specifications.

As NDN was not designed for wireless ad hoc networks, this thesis presents how interest and

[4] http://mininet.org/

[5] https://github.com/intrig-unicamp/mininet-wifi

data packets are treated by nMANET in order to address the limitations of MANETs. This thesis depicts the differences between the traditional NDN and the nMANET approaches.

Finally, the prototype of nMANET is compatible and interoperable with NDN platforms and implementations. Similarly, Mini-JNFD can execute both nMANET and NDN implementations.

1.6 Thesis outline

The rest of chapters of this thesis is organised as follows:

Chapter 2 - Humanitarian Mobile Technology This chapter presents the importance and limitations of mobile technologies in humanitarian relief scenarios and discusses reports from international organisations such as the United Nations that stress the urgent need to develop more people-centric approaches rather than traditional centralised technologies. It highlights the role that mobile communication and devices have played in a number of real emergency scenarios, such as the Haiti and Tohoku earthquakes, and their limitations in supporting humanitarian relief. As a consequence, this chapter introduces Mobile Ad Hoc Networks (MANETs) as a mobile communication technology that aims to provide decentralised data distribution in emergency scenarios. It discusses MANETs' existing challenges and limitations in real implementation platforms such as the Android operating system. The conclusions of this chapter are the motivation for Chapters three and four, where the concepts of Information Centric Networking and the named-based networking approach are presented and applied to address the challenges that MANETs still are facing in humanitarian relief.

Chapter 3 - Information Centric Networks Information Centric Network (ICN) is an alternative future Internet architecture where data is distributed based on names of content rather than IP host addresses between endpoints. This chapter presents ICN and compares the most relevant ICN projects to identify a suitable one to be applied in MANETs. As a result of these comparisons, this chapter concludes that the Named Data Networking (NDN) project is the instance that offers the most substantial academic and technical support, which is the main reason this chapter dedicates several subsections to explore the internals of the existing NDN implementation. This chapter is an important part of this thesis as it represents the conceptual background and base of nMANET, which is discussed in chapter four. The concept of an nMANET is inspired by NDN.

Chapter 4 - Name-based Mobile Ad Hoc Networks (nMANET) Chapter four presents the investigation of applying the NDN protocol in MANETs, which results in the creation of

the nMANET approach. The benefits of this approach ensure re-testing and reproducibility of experiments of NDN and MANETs. Subsequent discussions present the internal characteristics of nMANETs, which include how NDN packets are handled by nMANETs to achieve decentralised data distribution in ad hoc networks during an emergency scenario.

One of the important goals of this chapter is to show that NDN needs to be extended in order to be applied in MANETs. These extensions result in the creation of new strategies that nMANET offers to forward NDN packets from one node to others and to discover and retrieve content from information providers. It also concludes with additional features of nMANET that differ from the original NDN approach. This chapter is the foundation of the implemented prototype of an nMANET and the experimentation, which are presented in chapters five and six.

Chapter 5 - JNFD prototype design and implementation The software design, implementation and testing of the Java-based NDN Forwarder Daemon (JNFD) is described in this chapter. JNFD is an Android- and Linux-based service that complies with the nMANET requirements, software engineering principles and continuous delivery practices. This chapter describes the main modules of this prototype and its use during the experiments presented in chapter six. These modules include JNFD as a daemon service, Mini-JNFD as a virtual environment to simulate small and medium scale mobile ad hoc network topologies and the Test and Experimentation Manager as the main tool for continuous integration testing. Finally, additional information about how to incorporate new forwarding strategies is described.

Chapter 6 - JNFD framework validation This chapter presents the experiments using the nMANET implementation described in chapter five. This set of experiments are intended to show and validate that JNFD was tested in real devices such as Android smartphones and virtual environments through simulators such as Mini-JNFD. Each experiment includes a functional description of the experiment, the methodology utilised to collect data sets, and the analysis of the results. As each experiment contains a large amount of technical details that cannot all be included in this chapter, the thesis provides links to the Experimental Data Repository (EDR) that stores the collected data sets, the scripts utilised for data post-processing and descriptive statistical analysis.

Chapter 7 - Conclusions and future work A summary of the main conclusions of this thesis can be found in this chapter. It lists most relevant parts of this work, which become a motivation for future work in this research area.

CHAPTER TWO

MANETS IN HUMANITARIAN MOBILE COMMUNICATION

2.1 Introduction

Today's smartphones provide a wide range of resources and services, such as mobile applications that allow users to share content of mutual interest with peers in their proximity [Lane et al., 2010]. Furthermore, due to the integration of sensors, such as humidity, pollution, and blood pressure meters, mobile devices can support *context aware* applications that provide content not just in response to a specific user request but also taking into account the context in which a request is made or even pushing information to the user on the basis of the context alone [Yürür et al., 2016].

In 2011 the United Nations Office for the Coordination of Humanitarian Affairs (OCHA), reported that sharing content and resources, especially between mobile devices, plays a vital role in the coordination and exchange of information between victims and responders during emergencies [Crowley and Chan, 2011]. However, mobile technologies are limited in situations when the traditional (centralised) infrastructure becomes unavailable, for example, in the case of energy systems of cellular radio base stations being damaged by earthquakes. Consequently, victims get disconnected from centralised service providers and humanitarian relief. However, resources available in these disconnected communities can still be utilised for ad hoc communication and self-help. One approach that makes use of local mobile

resources and aims to link disconnected victims is Mobile Ad hoc Networks (MANETs) [Lien et al., 2009, Anjum et al., 2015].

This chapter first discusses the importance of applying mobile technologies in humanitarian relief by analysing real emergency scenarios and presenting an overview of trends in mobile devices. Next, MANETs are introduced as a mobile technology capable of empowering victims and responder during emergencies. Although decentralised MANETs have been thoroughly discussed in the literature, a number of issues remain to be tackled in real environments, for example, the efficient use of critical resources such as battery power remaining in mobile devices [Oh et al., 2010].

2.2 Importance of humanitarian mobile communication

According to the 2013 Annual Disaster Report of the United Nations(UN), after the 2011 tsunami affecting the north east coast of Japan, online connectivity became a luxury for victims within the disaster area in Tome, a local Japanese community. Keiichi Sato, the manager of the Tome community said:

> *"The Internet is useful for people outside of the disaster area, but inside the area, the power and network are often cut, so it's not useful right after the disaster, when we really need information. In addition, too much information can lead to confusion" (United Nations [Johnson, 2013], page 53).*

Mr. Sato helped victims get basic resources, such as water, food and electricity by broadcasting information to the community. In his statement, Mr Sato emphasised that victims are the main actors during and after an emergency scenario. He also raised the question of how portable devices may provide or distribute the necessary information to reduce risks for victims.

Natural disaster cases, such as the 2011 Tohoku earthquake in Japan, also show that the traditional communication infrastructure is vulnerable, and that mobile devices represent a potential lifeline infrastructure for ad-hoc and decentralised communication [1].

The 2013 UN annual report highlights the importance of mobile devices to victims and responders in the area of an emergency, and also to digital volunteers. All these actors agree that mobile devices, such as smartphones and tablets, have become an integral part of daily life and an important part in humanitarian relief [Crowley and Chan, 2011].

[1] https://tools.ietf.org/html/draft-irtf-icnrg-disaster-02

Figure 2.1 depicts three humanitarian mobile communication scenarios between responders, victims and outsiders. Firstly, in the centralised scenario, victims are completely dependent on remote responders and on Internet connectivity. Secondly, in the decentralised scenario, victims, although being disconnected from remote responders, can still communicate within the community in ad hoc mode. Finally, the hybrid approach merges both centralised and decentralised scenarios, and represents an ideal scenario allowing mobile victims, responders and outsiders to collaborate, with or without Internet access.

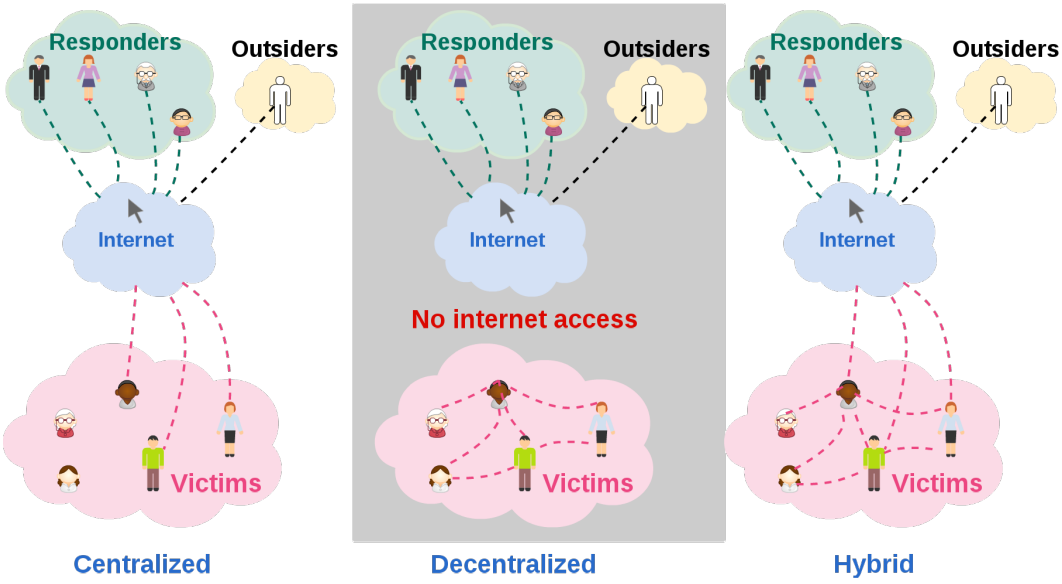


Figure 2.1: Humanitarian technology scenarios

As the right to receive and provide humanitarian assistance is a basic human right, worldwide humanitarian institutions, such as the United Nations, have concluded that empowering individuals and communities is a critical milestone in creating decentralised or people-centred approaches and increasing response capacity with self-organisation and mutual collaboration in emergency areas [Crowley and Chan, 2011]. The following presents cases of real emergency scenarios that show the importance of mobile applications and the need to empower victims through alternative decentralised mobile communication technologies, both to enhance humanitarian relief.

The following case is reported by Patrick Meier in his "Digital Humanitarians" book [Meier, 2015]. According to Meier, the telecommunication company Digicel Haiti Corporation launched the free 4346 SMS number to allow victims and responders to report the status of people and the affected area to a centralised Haiti Digital Crisis Map. The collected data was plotted in this map with the aim to identify the areas affected by the earthquake and to coordinate relief between victims and responders. The technology utilised included Open Street Map [2] for

[2] http://www.openstreetmap.org

mapping, the centralised response platform Ushahidi[3], as well as communications tools such as Skype [4] and social media (mainly Facebook [5] and Twitter[6]).

The results show that within days, dozens of Haitians from the capital were reporting on the devastation using tweets containing geographical clues to point to locations on the map. Through this, for example, the address of a pharmacy in the area of Petionville in Port-au-Prince that was still open and could supply first aid resources was identified.

It is evident that mobile devices can have an important impact on the way people communicate, collect and process information during and after disasters. In the case of the Haiti earthquake, the United States Federal Emergency Management Agency (FEMA)[7] and a Swedish Foundation, Flowminder[8], each used mobile applications to crowd-source pictures and locations during the disaster. Flowminder collected, aggregated and analysed anonymous data obtained from 1.9 million cell phones. The results showed that within three weeks of the disaster about 20% of the population in Port-au-Prince left the city.

However, these centralised communication services were not available immediately for victims in the area of emergency. For example, in the city of Port-au-Prince around 70 percent of the cell phone towers were destroyed by the earthquake. Consequently, mobile devices were often unable to establish connectivity to the central cellular switching platform to utilise mobile communication services, such as the free 4346 SMS service. The victims' high dependency on the availability of a centralised infrastructure shows the need to empower victims with alternative decentralised mobile communication technologies.

Mobile devices can play an important role in emergency situations, both during an acute crisis, in the immediate aftermath, and also in the longer-term, when managing the longer-term impact an emergency has. For example, Kaye et al. [Kaye et al., 2012] describe the use of near-field communication (NFC) in tracking chlorine usage in water treatment in Haiti. Also, their role is not limited to what might traditionally be understood as emergency relief but extends into all areas of maintaining or re-establishing a degree of normality in the social life of a society in the face of adverse events. For example, in a region in Kenya affected by drought, mobile phones were used to transfer cash between subscribers. The scheme, run by the World Food Programme and Orange, involved more than 7000 people in the town of Isiolo[9] [Drummond and Crawford, 2014].

[3] https://www.ushahidi.com/
[4] https://www.skype.com
[5] https://www.facebook.com/
[6] https://twitter.com/
[7] https://www.fema.gov/
[8] http://www.flowminder.org/
[9] https://www.wfp.org/stories/mobile-cash-emergency-response

There is a general shift in emphasis from emergency relief to *preparedness*, as illustrated also by a preparation mobile application for hurricanes, earthquakes, tornadoes and wildfires launched by the American Red Cross. Using this application before and after Hurricane Sandy, victims were able to check in advance for shelters, food, water and government relief locations, which also underlines the importance of mobile devices and mobile applications in disaster scenarios [Johnson, 2013].

Other application areas for mobiles include tele-health applications such as the mobile platform described by Wouhaybi et al. [Wouhaybi et al., 2013] that can be used to capture and record physiological and audio context from patients and can be accessed locally or remotely.

From the digital humanitarian point of view, another application of mobile devices lies in the area of social networks. According to Al-akkad et al., mobile networks, and in particular mobile applications and sensors are able to collect and handle data based on two main approaches [Al-Akkad and Zimmermann, 2011]. One is referred to as *participatory sensing* where data is collected in collaboration with the owner of the device. On the other hand, *opportunistic sensing* involves autonomous data collection. Social network applications follow both of the above sensing methods and they successfully assist owners of mobile devices to crowd-source data in emergencies. An example of the observations of Al-akkad et al. is the case of the 2011 Tohoku earthquake in Japan [Johnson, 2013], when the Japanese social media intensively used data crowd-sourcing on mobile devices and social networks, such as Twitter, by sharing pictures taken by locals during and after the natural disaster.

Additionally, and from the victim's perspectives, Al-Akkad et al. conducted direct interviews to understand people's acceptance of mobile devices in emergency scenarios. The general conclusion was that people firstly tend to help victims and look for surrounding relief units. In case they do not find support, they make emergency calls. According to the study, victims may use an emergency mobile application as long as the application is straightforward, for example, if it helps perform tasks within a short time, and if it is free of charge[Al-Akkad and Zimmermann, 2011].

2.3 Limitations of humanitarian mobile technology

Despite the effort international humanitarian organisations have invested in mobile communication technologies and the research done in this area, numerous limitations remain. These include a high dependency on a centralised infrastructure, which may potentially cause communication bottlenecks.

This limitation is highlighted by the Haiti earthquake, where, as responders and victims utilised

the Digicel SMS 4346 service, they were entirely dependent on its availability. Additionally, the service may not have covered all potential areas where victims were located. For example, in the city of Port-au-Prince around 70 percent of the cell phone towers were destroyed by the earthquake [Crowley and Chan, 2011]. Furthermore, digital volunteers outside the emergency area had to address challenges arising in such conditions, which included translating between the local Haitian Creole language and English as well as analysing the content of messages to respond to the most urgent and life-threatening cases.

According to the analysis of the Haiti earthquake by Flowminders, estimating internal migration or population movements accurately during the earthquake using mobile phone data obtained from service providers was only feasible in areas characterised by a high use of centralised mobile services, leaving the rest of the community invisible.

The second limitation arises from the lack of Internet access. For example, the videos that were created and shared by the Haiti Crisis Map team and posted in centralised systems such as Youtube, were only available to those with Internet access. This lack of Internet connectivity was also exemplified by the case of the Tohoku earthquake in Japan. The first generation of web-based emergency assistance platforms, such as Ushahidi[10], Site-Seeing [Hughes and Palen, 2009] and Google People Finder[11] also assume that the Internet connectivity between victims, responders or bystanders is of sufficient quality, and users possess the required skills to handle the data. Such assumptions cannot be satisfied in many cases.

2.3.1 Humanitarian relief challenges in communication

Considering the emergency scenarios of Section 2.2, communication in humanitarian relief faces numerous challenges, such as the ones described in the 2017 Internet-draft by the Information-Centric Networking Research Group[12] on “Research Directions for Using ICN in Disaster Scenarios”.

These challenges include enabling usage of functional modules of a fragmented infrastructure to provide some level of service even as they are disconnected from the wider network.

Decentralised user authentication and trust arise as another challenge as access to centralised authentication and trust mechanisms is lost when networks are partitioned.

Additionally, due to problems such as broken cables and failed routers, the capacity of the overall communication capacity is diminished, which increases the possibility of congestion, therefore

[10] http://www.bu.edu/bostonia/summer13/damiano/
[11] https://google.org/personfinder/global/home.html
[12] https://tools.ietf.org/html/draft-irtf-icnrg-disaster-02

delivery of humanitarian information to victims and responders cannot be guaranteed. This also means that this fragmented infrastructure cannot support end-to-end delivery of humanitarian information.

One of the most important challenges is energy consumption. Power outages and the remaining energy in batteries define the lifetime of the communication infrastructure and its overall utility. Consequently, energy efficient approaches are imperative to be considered in humanitarian communication.

2.3.2 Decentralised communication as an alternative

According to the 2013 World Disaster Report of the International Federation of the Red Cross [Johnson, 2013], most lives are saved by local responders and by using local resources. For example, in emergencies people do not only share or donate available resources, such as water, food or shelter, they also donate and share available digital data including pictures with geographical locations posted on social media [Meier, 2015].

Based on experiences from international humanitarian institutions, victims, responders and digital humanitarians convey the message that the best way that humanitarian technologies can empower people is by providing them with decentralised communication using mobile devices. Note that mobile devices play a significant role already in addressing disasters, as they cover the gap between responders and victims to exchange information. However, this is currently possible only in cases when the network infrastructure is available [Johnson, 2013].

Although responders and outsiders are important actors during an emergency, victims play a critical role during a crisis and afterwards during the recovery process. In fact, one might argue that the distinction between responder and victim is of limited. Local residents have a better understanding of the affected area and resources, speak the local language, and have developed social relationships prior to the emergency. Consequently, aid agencies often draw on the local population to source the human labour required for relief operations. This is in addition to the relief work that people affected would do anyway.

The importance of empowering victims with decentralised mobile technologies can be exemplified by illustrating how mobile ad-hoc communication could aid victims in cases such as the Haiti Earthquake. In this case, volunteers utilised social media such as Facebook, Twitter and Skype to build the Haiti Crisis Map, which was used to connect responders with those victims who were lucky enough to have access to a centralised infrastructure that was still functional.

In areas where victims were completely or partially disconnected, mobile ad-hoc communication

could stablish connectivity between them to build a decentralised or hybrid communication infrastructure as depicted in Figure 2.1. Therefore, the Haiti Crisis Map would not be restricted to people with Internet connectivity only; victims and responders without Internet access could also collaborate if they were connected in ad-hoc mode and utilised suitable applications. Providing ad-hoc mobile communication technologies to victims allows them to become first responders on site and first digital responders on- and offline.

2.4 Trends in mobile device capabilities

The International Telecommunication Union (ITU) reports that in 2017 the number of mobile cellular subscriptions has reached an estimated 7.74 billion (cf. Figures 2.3 and 2.2), which is more than the world population [ITU, 2017b]. What is even more remarkable is the growth in mobile broadband subscriptions. Just as mobile phones are increasingly replacing fixed telephone lines, mobile broadband is growing much faster than fixed line broadband and now stands at 4.22 billion, Figure 2.4. The global penetration rate of mobile broadband is estimated to have risen to 56.4 percent.

The ITU report also highlights that mobile broadband penetration in Africa remains low at 26% in contrast to Europe and the Americas, where mobile broadband penetration has reached 85.2% and 86.3%, respectively [ITU, 2017a]. These percentages show that there are still people and communities around the world who are invisible and not connected to the available digital societies and resources around the world. Potential reasons may include elevated broadband service prices, as the price for mobile broadband services in developing countries is higher than in developed countries. For these populations, being disconnected from the global Internet is not something they experience only in crisis situations but part of their everyday normal lives.

These trends show that mobile devices are rapidly changing and their presence is increasing all over the world, Figure 2.5. Market research company IDC reports [13] that companies shipped 344.3 million mobile phones in the first quarter of 2017. The most popular mobile platforms were Android [14] and Apple iOS[15], reaching 85% and 14.7% of the total mobile phone shipments, respectively.

[13] https://www.idc.com/promo/smartphone-market-share/os
[14] https://www.android.com/
[15] https://developer.apple.com/ios

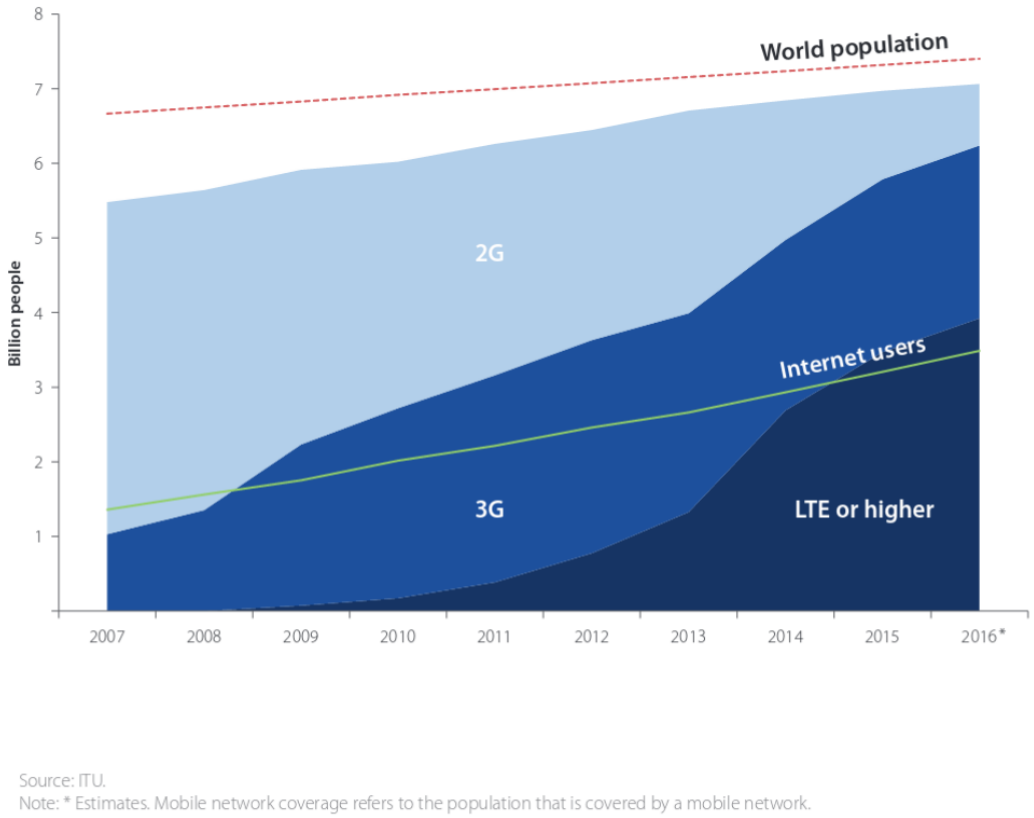


Figure 2.2: Mobile network coverage and trends

2.5 Mobile Ad-hoc communication

As mentioned in section 2.3, in the case that people are completely disconnected, Mobile Ad-hoc Networks (MANETs) represent an alternative technology to empower victims and responder for humanitarian coordination. This section introduces MANETs, their characteristics and limitations. It also provides classifications for energy-aware routing protocols, which are assessed based on their implementation and testing in real mobile infrastructures. Finally, this section concludes with the challenges MANETs still face.

2.5.1 Mobile ad-hoc networks

A Mobile Ad-hoc Network (MANET) is a group of at least two autonomous mobile devices capable of establishing connectivity with each other and communicating with other nearby devices [Murthy and Manoj, 2004]. Members of a MANET self-organise, self-configure, and collaborate with each other to establish communication by each forwarding packets for others.

Unlike traditional centralised platforms, such as cellular networks, MANETs do not depend on a

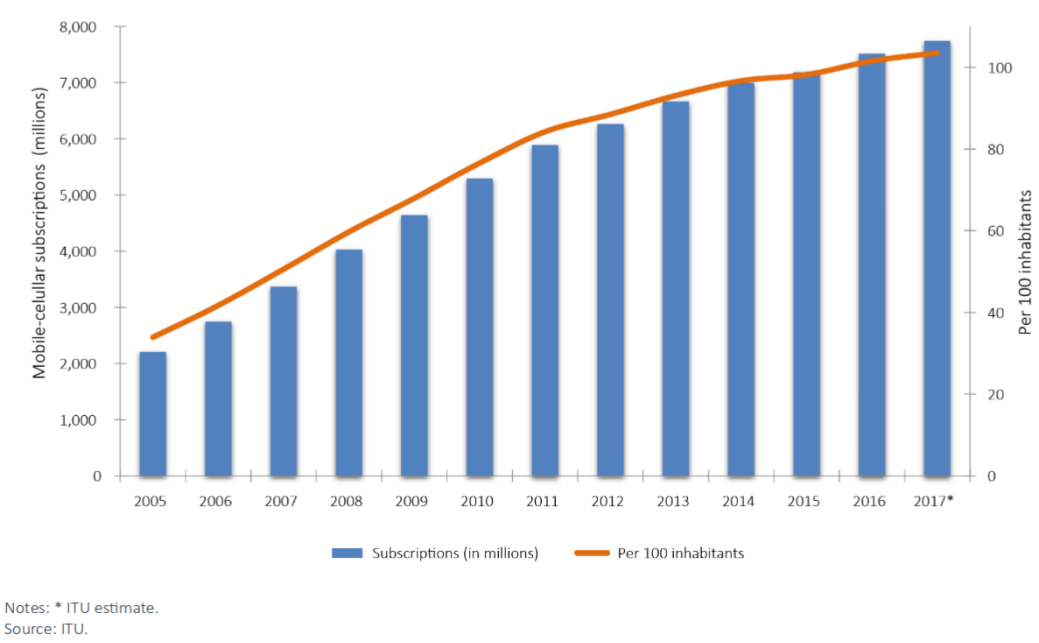


Figure 2.3: Global Mobile cellular subscriptions according to ITU.

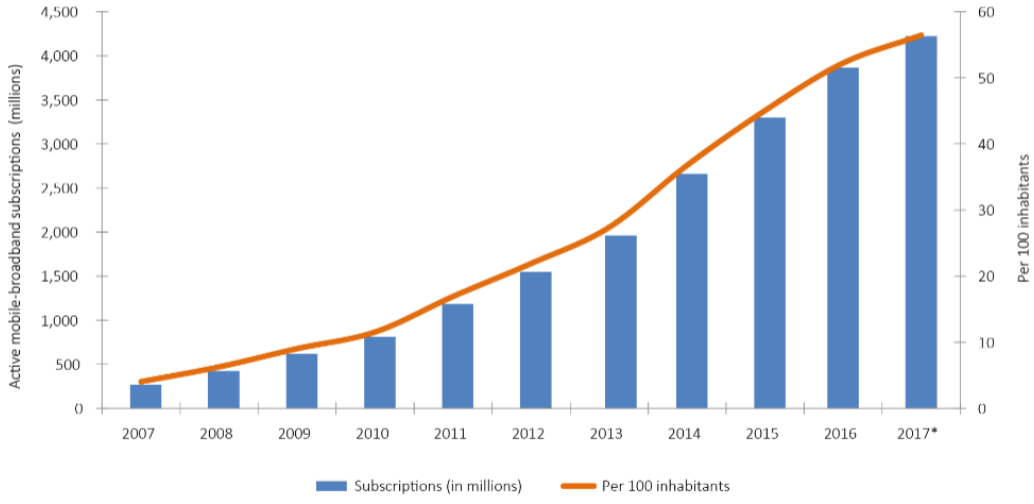


Figure 2.4: Global Mobile broadband subscriptions according to ITU.

centrally administered infrastructure. In cellular networks, radio base stations and a centralised switching centre are utilised to make routing decisions between end points. On the other hand, a MANET requires nodes to make routing decisions and resource management completely among themselves, which increases its complexity [Murthy and Manoj, 2004]. The summary in Table 2.1 shows the major differences between cellular networks and MANETs.

By establishing ad-hoc communication between participating mobile devices and by combining

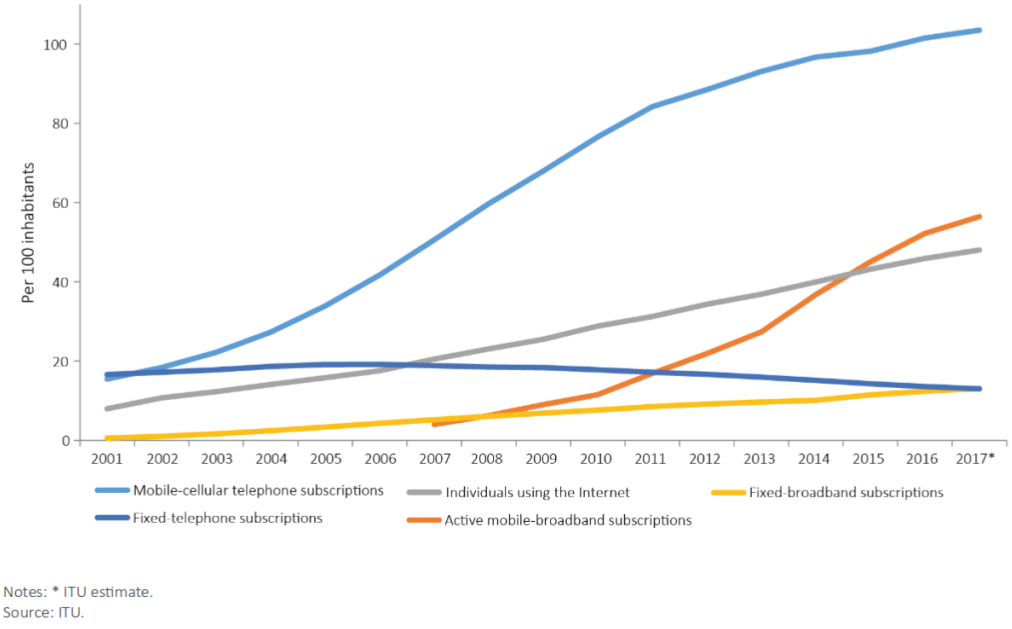


Figure 2.5: Growth of Information and Communication Technologies.

Cellular Networks	**MANETs**
Infrastructure-based	Infrastructure-less
Single hop wireless links	Multi hop wireless links
Guaranteed bandwidth	Shared radio channel
Centralised routing	Distributed routing
Seamless connectivity	Frequent path breaks due to mobility
High cost and time for deployment	Quick and cost-effective deployment
High cost of network maintenance	Self-organisation and self-maintenance
Mobile hosts require low complexity	Mobile hosts are required to make complex decisions.

Table 2.1: Major differences between cellular networks and MANETs [Murthy and Manoj, 2004]

and aggregating their resources, MANETs enable a range of applications. The fundamental capability required for these applications is to effectively and efficiently move data between the nodes in the network. Based on this capability, applications can be built that enable collaboration services and allow data to be collected and processed.

2.5.2 Main characteristics and issues in MANETs

The following paragraphs list the main characteristics of MANETs to be considered at the stage of design, implementation, deployment and maintenance [Murthy and Manoj, 2004, Roy, 2010].

Medium Access Control Wireless interfaces of nodes utilise transmission channels as a common radio frequency space for transmitting and receiving packets. Due to the nature of broadcasting in radio environments, two or more nodes cannot transmit simultaneously if they want to avoid transmission collisions. Medium Access Control (MAC) is the mechanism needed to share this radio frequency space efficiently between nodes of a MANET. Table 2.2 list major issues to be considered in the design of a MAC protocol.

Issue	Description
Distributed operation	The MAC protocol design should involve minimum control overhead and be fully distributed.
Synchronisation	The design should include synchronisation with minimum resource utilisation such as bandwidth and battery power.
Hidden terminals	Hidden terminals are nodes unreachable from the sender; however, they are reachable by the receiver. Hidden terminals reduce the throughput of a MAC protocol used in MANETs.
Throughput	MAC protocols should maximise the throughput, which means minimising collisions and maximising channel utilisation.
Delay	This represents the average time that a packet spends in transmission and it should be minimised.
Fairness	This means the ability of the MAC protocol to share equally the bandwidth between all nodes. An unfair approach results in draining the resources of a node much faster than the other nodes.
Real time traffic	Real time traffic such as voice, and video conference requires explicit MAC protocol support.
Resource availability	The MAC protocol should be able to estimate the resource availability on each node, such as bandwidth.
Power control	It should optimise the use of energy, decrease interference with neighbouring nodes and increases frequency reuse.
Adaptive rate control	MAC protocols should use high data rate transmissions when nodes are nearby, and it should be adaptively reduced as the nodes get far away from each other.

Table 2.2: Issues of MAC protocol design and implementation [Murthy and Manoj, 2004]

Routing The decentralised nature of MANETs, and the unpredictability and high mobility of their members cause changes in the network topology over time. Therefore, establishing

communication between two members of the network requires a routing protocol to define a path between both endpoints, and also update routing information between them. The criteria to find and update routing paths include the hop length, the lifetime of wireless links, broken paths, and the utilisation of the available bandwidth. Table 2.3 summarises the main issues that routing protocols face in MANETs, and Table 2.4 presents the characteristics required by routing protocols in MANETs.

Issue	**Description**
Mobility	Mobility of nodes produces changes in the state of links, packet collision, loops, and resource availability.
Bandwidth constraints	The available bandwidth per node depends of the number of nodes sharing the channel and the traffic generated by them.
Network load distribution	A routing protocol should offer a mechanism to uniformly distribute the network load in the MANET.
Other constraints	Capability of routing protocols are influenced by resources such as remaining power, and available buffer space.

Table 2.3: Issues of routing protocols in MANETs [Murthy and Manoj, 2004]

Broadcasting Broadcast delivers packets from one sender to all nodes that are located within the sender's transmission range area. Broadcast allows to disseminate packets in a one to many manner. Broadcast has useful application in MANETs, such as neighbour discovery and location, route establishment. It is a fundamental operation due to the nature of radio transmission broadcasting. However, broadcasting is not reliable due to the lack of acknowledgement and the possible interferences and collisions between sender and receivers [Roy, 2010].

Multicasting Multicast delivers a data packet from one sender to a set of destinations that belong to a group, enabling point-to-multipoint and multipoint-to-multipoint communication. In wired networks, multicasting is enabled by the routing infrastructure in networks that support multicast. In MANETs, implementing multicast is a challenging problem due to the mobility and the constraints in resources of nodes.

Transport layer protocols Transport layer protocols are responsible for establishing and maintaining end to end connections, reliable data packet delivery, flow control and congestion control [Kurose and Ross, 2013]. Due to the mobility of nodes, establishing effective end-to-end communication between nodes in the MANET is a challenging task. Therefore, traditional protocols such as the *Transmission Control Protocol (TCP)* and the *User Datagram Protocol (UDP)* do not work as well in MANETs as they do in wired networks or infrastructure mode wireless networks.

Issue	Description
Path finding delay	The time spent to find a path between two nodes should be as small as possible.
Quick reconfiguration	Routing protocols should be able to quickly reconfigure paths in order to handle packet losses due to path breaks and changes in the topology of the MANET.
Loop free	Routing protocols should be able to detect and handle loops. Loops may form in the route due to the mobility of nodes. Loop-free routing is a fundamental requirement that avoids waste of network resources such as bandwidth.
Distributed routing	The routing protocol should be fully distributed as centralised approaches lead to consuming more resources such as bandwidth.
Minimum control overhead	Control packets exchanged between nodes to find new paths and update existing ones should be kept to a minimum. Control packets consume bandwidth and can reduce throughput due to collisions with data packets.
Scalability	The routing protocol should scale well to a large number of nodes.
Quality of service(QoS)	Routing protocols should provide a QoS level demanded by nodes. QoS parameters can be bandwidth, delay, jitter, packet delivery ratio, and throughput.
Real time traffic	Routing protocols should be able to support real-time traffic such as in the case of video conferences.
Security	The routing protocol should be resilient to attacks.

Table 2.4: Requirements of routing protocols in MANETs [Jacquet et al., 2001, Murthy and Manoj, 2004]

TCP TCP is a connection-oriented protocol, which means that prior to the data exchange between source and destination, an end to end connection is established. TCP breaks down the stream of bytes into segments that are reassembled in correct order at the receiving endpoint and retransmitted in case they were lost. TCP performance is affected significantly by node mobility as it causes frequent path breaks, nodes with stale routing information, high channel error rates and frequent network partitions. When a path breaks, the process of reconfiguration depends on the utilised routing protocol. The process of finding a new alternative path may cause retransmissions as well as causing congestion algorithms to limit the sending rate, which combine to degrade throughput.

UDP Unlike TCP, UDP does not require a pre-established connection between the end points. It is a connectionless transport protocol that neither guarantees reliable packet delivery nor executes congestion- or flow control. UDP does not take into account the status of the links in the network and also does not offer error correction and retransmission mechanisms. Its ‘fire and

forget' approach is not affected by node mobility in the same way as TCP is, though applications such as voice communication may suffer degradation of quality due to lost packets.

Volunteer participation Endpoints in a MANET rely on the availability of intermediate nodes and on the willingness of their owners to contribute their resources to the operation of the MANET. At time when resources are scarce, such as when battery power runs out in an emergency situation, participants may decide to withdraw. While incentive schemes may work in general, it is not clear that they would in crisis situations.

Self-organisation Self-organisation is an important characteristic of MANETs. Nodes need to discover their neighbours and learn about their place in the overall topology as well as constantly update this information. To obtain and maintain the status of neighbour nodes requires periodic control packet exchanges. Due to the mobility of nodes, topology reorganisation implies gathering information about the status of nodes which may include failure of nodes such as when their batteries become depleted. MANETs should be able to perform self-organisation quickly, efficiently and the process should be transparent to the end user.

Address and service discovery Addresses or unique identifiers are required by a node to participate in a MANET running a connection-oriented protocol stack such as TCP/IP. Therefore, a mechanism for auto-configuration of addresses is required to assign unique addresses to each node. Uniqueness needs to be guaranteed as network nodes join or leave the MANET and as networks are partitioned or merged. Additionally, a mechanism to discover services is also required. This mechanism should allow nodes to discover and locate services offered by other nodes.

Quality of service Quality of service (QoS) is the level of performance offered by the network. Although the type of parameter utilised to measure QoS depends on the type of application, such as availability and minimum energy consumption in emergency scenarios, for routing protocols the QoS parameters include network throughput, packet delivery ratio, reliability, delay, delay jitter, packet lost rate, bit error rate, and path loss.

Energy As nodes in a MANET are mobile devices, battery life constrains the lifetime of a node and of the network as a whole. The energy consumption of wireless communication is a significant factor in overall energy consumption and is dependent on factors such as the state of operation that include transmit, receive or sleep mode as well as the technology of the radio circuitry and the transmission power level set. In real scenarios, it is difficult to replace or recharge batteries while the device is moving and without interrupting the communication.

Therefore, to maximise the network lifetime, the efficient use of the energy stored in batteries becomes a key metric for MANETs.

Security Nodes in a MANET are vulnerable to attacks due to the lack of a central authority for coordination and because nodes share the wireless medium access space. These attacks are generally classified into two types: passive and active. The passive attacks refer to the situations when the attacker collects information from the network without disturbing its operation. On the other hand, in addition to obtaining information, active attacks disturb the network by using aj node that can be inside (internal attack) or outside the network (external attack). Table 2.5 summarises common threats in MANETs.

Attack	Description
Denial of service	An attempt to make the network and/or nodes unavailable by disrupting their operation or overloading them.
Resource consumption	This attack consumes network resources, principally by depleting the battery power to critical levels, and by overflowing buffers with unwanted information, such as filling routing entries with unwanted or non-existing destinations.
Host impersonation	A node act j as another node and responds with correct control packets to create wrong routing entries, and even stops the traffic to a destination.
Information disclosure	Causing the release of confidential information to an unauthorised node.
Interference	When a wide spectrum noise is injected in the wireless communication channel, which can be achieved for example by sweeping across the spectrum.

Table 2.5: Common threats in MANETs

2.5.3 Routing protocols classifications

Routing protocols in MANETs should handle changes in the network topology caused by the mobile devices joining and leaving the network at any time and moving to different places in the topology. Additionally, routing protocols should optimise the use of local resources, such as the remaining energy of batteries. Routing approaches with excessive overhead traffic are not suitable due to the volume of network traffic they produce and their consumption of available resources, especially in large networks.

The literature on MANETs has classified routing protocols from different perspectives [Roy, 2010, Loo et al., 2011, Boukerche et al., 2011]. As there is no generally accepted classification, out of the various classification approaches [Loo et al., 2011], this section adopts and

presents a broad classification based on the mechanism taken to update routing information. Routing protocols are categorised into the following three groups: proactive, reactive and hybrid [Murthy and Manoj, 2004].

2.5.3.1 Proactive

Nodes using a proactive routing protocol maintain an up-to-date routing table containing routes to every other node in a MANET by flooding routing information to the entire network, even if the routing information is not required. Routing tables have to be periodically updated to reflect network topology changes, so nodes must continuously find routes to all possible and reachable destinations. Such proactive protocols may cause significant routing overhead and, consequently, increased bandwidth usage and a higher energy consumption. A well-known example of a proactive protocol is the Optimized Link State Routing (OLSR) protocol [Jacquet et al., 2001, Clausen and Jacquet, 2003], which is described in the following.

Optimized Link State Routing (OSLR) As proactive routing protocols periodically update routing information between nodes, OLSR periodically floods the link state of nodes in an optimised manner to preserve the use of resources such as bandwidth. OLSR, which is described in RFC 3626 [Clausen and Jacquet, 2003], optimises the diffusion of routing information by using a mechanism called MultiPoint Relaying (MPR) and the main control messages of which are summarised in Table 2.6.

Control message	Description
HELLO	Message sent to all neighbours and used for neighbour discovery and MPR calculations.
Topology Control (TC)	Message that includes the link state information.
Multiple Interface Declaration (MID)	Contains the list of IP addresses used by a node.

Table 2.6: OLSR control messages

Nodes discover their one-hop neighbours by transmitting HELLO messages. From these one-hop neighbours, OSLR selects a set of nodes for forwarding purposes and calls them MPRs. MPRs are nodes allowed to only forward partial routing information about the network. This first optimisation minimises the number of retransmissions in the network. A second optimisation is the reduction of the size of the control messages by selecting a subset of link states for updates, which minimises the overhead [Boukerche et al., 2011, Murthy and Manoj, 2004].

Finally, nodes that are not MPRs can receive updates; however, they do not forward them. MPRs should be selected in a way to cover all two-hop neighbours, while broadcasting the least number of packets [Loo et al., 2011, p.26].

2.5.3.2 Reactive

In contrast to proactive protocols, reactive protocols do not exchange routing information periodically; instead, they update routing information when it is needed at the time of communication. By using a reactive or on-demand protocol, the node initially searches for a path to the destination and then establishes the connection to execute the data transfer. As a result, the final routing table contains information only about active routes. Although reactive protocols may lead to long delays by searching routes, they cause less overhead and tend to consume less bandwidth and energy than traditional proactive protocols. The following paragraphs describe a well known reactive protocol, Ad-hoc On-demand Distance Vector.

Adhoc On Demand Distance Vector (AODV) AODV is a reactive protocol that is described in RFC 3561 [Perkins et al., 2003]. To find the path to a node which has an unknown route, a node starts the path discovery process by flooding a route request (RREQ) control message to neighbours, and these nodes broadcast further to nearby nodes until the message reaches the destination itself or an intermediate node with a valid route to the destination. Intermediate nodes have a record of the nodes from where the first copy of the RREQ has arrived. These records are used to build the reverse path when the destination replies to the origin with an RREP message, described in Table 2.7.

In the case that a link between the origin and destination of the found path breaks, the adjacent nodes to the broken link send a route error (RER) control message to end nodes to inform about this modification. Thus, end nodes remove the broken link in their tables, and the originator node re-starts a new discovery process to the destination.

Control message	Description
RREQ	A route request message is used when it is required to find the path to another node.
RREP	A route reply message is sent using the unicast reverse path until reaching the originator of a RREQ.
RERR	A route error message is used to notify other nodes of a link breakage.

Table 2.7: AODV control messages

One of the disadvantages of AODV is the case where intermediate nodes have stale routing information, which can lead to inconsistent routing. A second disadvantage is that multiple RREP message to a single RREQ control message can provoke overhead in the network, therefore leading to unnecessary bandwidth consumption.

2.5.3.3 Hybrid

Hybrid protocols are a result of combining proactive and reactive protocols, reducing their disadvantages and the overhead associated with route discovery. Most hybrid protocols are applied in networks that are divided by zones in a hierarchical configuration. In such networks, nodes that belong to the same zone proactively keep up-to-date routing tables to neighbours. This adoption is important as it provides a starting point to evaluate paths to nodes outside the zone by using a discovery strategy. Hybrid protocols eliminate the need to keep up-to-date routing information of the network. Nodes that are located within the same zone use proactive mechanisms; however, they follow a reactive routing behaviour to reach faraway nodes beyond this zone. An example which efficiently combines the best features of proactive and reactive mechanisms is the Zone Routing Protocol, which is described as follows.

Zone Routing Protocol (ZRP) The Zone Routing Protocol (ZRP) [Haas, 1997] is a hybrid protocol that selects a set of neighbours of the node of origin to form a zone. This zone is called an "intra-zone" and has a size (zone radius hop) that is defined by the number of hops from the node of origin. Nodes that belong to the intra-zone utilise proactive protocols and nodes beyond this zone utilise reactive ones.

To find a route to an unknown destination, the origin starts to discover neighbours either by utilising an Intra-zone Routing Protocol (IARP[16]). Once the request reaches the nodes on the border of the intra-zone (peripheral nodes) and in the case that the destination has not been reached yet, ZRP utilises the Inter-zone Routing Protocol (IERP[17]) to find paths outside the intra-zone. Additionally, ZRP utilises a mechanism called the Bordercast Resolution Protocol (BRP[18]) to control the traffic between zones.

Nodes that forward the route request packet (RREQ) also append their own address to the packet. Later, these addresses are utilised to deliver the route reply packet (RREP) to the origin (reverse path). In the case that a broken link is detected, an alternative path reconfiguration is performed and routing updates are sent to the origin to inform it about the routing changes.

[16] https://tools.ietf.org/id/draft-ietf-manet-zone-iarp-01.txt
[17] https://tools.ietf.org/html/draft-ietf-manet-zone-ierp-02
[18] https://tools.ietf.org/id/draft-ietf-manet-zone-brp-01.txt

One disadvantage of ZRP is the impact in the performance of the network due to the selection of the zone radius hop value. Additionally, ZRP tends to produce high overhead due to the large overlapping of intra-zones.

2.5.3.4 Energy-aware routing protocols

Mobile nodes in a MANET are powered by a limited supply of energy stored in their batteries. The efficient use of this remaining energy is an important feature in MANETs, as it has a direct impact on the network lifetime. Energy-aware routing protocols aim to minimise the power consumption of nodes and to maximise the network lifetime. Table 2.8 presents a summary of energy-aware routing protocols for different approaches [Mishra and Pattanayak, 2013]. These approaches are expanded upon in the following paragraphs.

Approach	Criteria of the best route
Transmission power	The route that minimises the total transmission power between endpoints.
Load distribution	The route that make uses of under-utilised nodes.
Power management	By turning off the power unit of unused intermediate nodes.
Sleep power down mode	By putting in sleep mode nearby nodes.

Table 2.8: Energy aware approaches and routing protocols [Mishra and Pattanayak, 2013]

Transmission power As the wireless area of coverage has direct relation to the transmission power of a mobile device, this approach considers an optimal routing path to be the one that minimises the total transmission power between source and destination. For example, the Online Max Min protocol(OMM) estimates the total power of routes between the origin and the destination, and selects from them the one that has the minimum value. Additionally, it selects alternative routes as those with a power consumption that deviates minimally from the optimal value.

The advantages of these type of protocols are that they find and utilise routes that consume minimum power. However, one of the disadvantages is that this approach is based on global energy information. It is required to know transmission power of all nodes in the network in order to find the best route, which implies transmission overhead due to the need for prior exchange of transmission power measurements.

Load distribution This approach considers the best routes to be those using nodes that have high levels of energy in batteries or nodes that are underutilised. In other words, it distributes load

to nodes with high levels of power in batteries. However, this could result in longer routes, as the packets are only forwarded by nodes with high values of remaining energy in their batteries.

An example of this type of protocol is the Localised Energy Aware Routing (LEAR) where a route request message is broadcast from the origin in order to find a path to the destination. Each intermediate node forwards the received route request message only when its remaining battery level is higher than a predefined threshold level, otherwise the message is dropped.

Therefore, as the destination node receives the route request message from intermediate nodes which level of battery are higher than a predefined threshold, a shortcoming resides in the complicated design of this threshold. This is because it is not fixed and must be be communicated among the nodes in the network.

Power management Protocols belonging to this approach minimise energy consumption by minimising the interference and by adjusting the transmission power of each mobile device in order to reach only a subset of nodes in the network. Therefore, the connectivity range changes dynamically in order to achieve the most suitable path between source and destination.

The Power Aware Multi Access protocol (PAMAS) is an example that uses this type of approach. PAMAS saves energy of mobile devices by turning off the power unit when the node is not in use. This includes the cases when a node can not receive or transmit packets. Additionally, it avoids the use of nodes with low levels of remaining battery. The disadvantage of this approach is that broadcasting can generate collisions with other messages in the intermediate receiver. Additionally, when the power unit of a node is turned off, it can cause the isolation of one-hop nearby nodes.

Sleep power down mode This approach utilises the radio hardware feature of mobile devices to put the radio subsystem into sleep mode or simply turn it off to save energy. A subset of nodes are put into sleep mode, and allowed to waken periodically to request updates to those who are permanently with their radio subsystem in idle state, called the master nodes.

This master-slave architecture can be exemplified by the SPAN protocol. In SPAN, master nodes provide the routing backbone and control traffic as well as channel contention. Nodes take the role of master when nearby nodes can not reach each other. This is a dynamic configuration, therefore does not provide a minimum number of masters. In this configuration masters are easily overloaded as this approach does not consider the levels of remaining energy in master nodes.

2.5.4 Real implementation

As the discussion in the previous paragraphs shows, various routing protocols for MANETs were proposed; however, there are very few applications that implement them in real scenarios, and even fewer in real mobile platforms. As described in section 2.5.1, there are various hurdles when it comes to deploying a MANET such as the mobility of nodes. This section describes deployments in real mobile devices based on the availability of their open source code, their available and up-to-date documentation and test results on real mobile platforms such as Android.

SPAN The Smart Phone Ad-hoc Network (SPAN) project [Thomas et al., 2012] is an open source implementation that links mobile devices in Ad-hoc mode. It is a "proof of concept" of a functional implementation for Android mobile devices. The application can detect nearby devices and utilises OLSR as its main routing protocol. Unfortunately, SPAN does not offer relevant documentation, and its code repository is not maintained, such that the reverse engineering would be required to uncover the internal design of SPAN.

BATMAN The "Better Approach To Mobile Ad-hoc Networking" (BATMAN) [19] [20] is a proactive routing protocol for MANETs. Nodes in BATMAN are not aware of all destinations across the network. Instead, each node perceives and maintains routing information about the next best hop only [Abolhasan et al., 2009].

In BATMAN, each node broadcasts an Originator Message (OGM) to neighbouring nodes to announce its existence, and these neighbour nodes rebroadcast this OGM a maximum of one time to their respective neighbours in a way that the network is flooded with OGMs.

To select the best next hop towards a destination, a node count the number of OGM received from other nodes in the network, and defines the quality of the respective link by the number of OGM received through that given link. Therefore, the best next hop is selected as the one from where a node has received the largest number of OGMs[21].

In general, implementations in real devices require the device to be rooted in order to be able to switch to Ad-hoc mode. This may partly explain why, despite significant research interest, few practically usable MANET implementations are available.

[19] http://www.open-mesh.org/projects/open-mesh/wiki
[20] https://wiki.freifunk.net/Glossar#B.A.T.M.A.N.
[21] https://tools.ietf.org/html/draft-openmesh-b-a-t-m-a-n-00

SERVAL Serval [Gardner-Stephen et al., 2017] [22] is an open source project [23] that provides free peer-to-peer mobile communication, SMS and file sharing over Wi-Fi. The motivation of this project is to provide a basic infrastructure-less communication tool for emergency scenarios and natural disasters.

Similarly to SPAN, the limitation of this project is that Serval requires root permission to be functional in Android mobile devices. This is because the Android operation system does not support an Application Programming Interface (API) for Ad-hoc mode. Additionally, mobile devices require administrative permissions to use the Ad-hoc mode of the Wi-Fi chipset in the mobile device, and also it assumes that the technology of the Wi-Fi chipset supports ad-hoc mode [Gardner-Stephen and Palaniswamy, 2011].

Firechat Firechat [24] is an IP based peer-to-peer application that lets Android mobile devices exchange information each other through Wi-Fi or Bluetooth. Firechat is developed by Open Garden [25], a company that maintains and provides Firechat products, such as free or paid broadband services [26]. This application gained popularity in social conflicts such as the protests of 2014 in Hong Kong, where protesters were able to communicate each other "off the grid" [27].

Firechat utilises the Open Garden communication protocol, which still is not open source. At the time of writing, no academic or official technical information could be found that describes the internals of this protocol and its implementation. However, it is possible to find unofficial reversed engineering attempts to describe the internals of this protocol [28]. This limitation minimises the applicability and extension of the Open Garden protocol to other applications or development.

2.6 Chapter summary

This chapter emphasises the importance of mobile communication as a technology that can empower victims and responders before, during and after an emergency scenario. International Organisations such as the UN, concluded that creating people-centred approaches is critical milestones to increase response capacity and straightforward collaboration between victims, responders and digital humanitarians.

[22] http://www.servalproject.org/
[23] https://github.com/servalproject
[24] https://www.opengarden.com/how-to.html?lang=en
[25] https://www.opengarden.com/team.html
[26] https://www.opengarden.com/protocol.html
[27] https://www.bbc.co.uk/news/av/technology-29448739/hong-kong-protesters-use-firechat-daisy-chain-app
[28] https://citizenlab.ca/2014/07/iraq-information-controls-update-analyzing-internet-filtering-mobile-apps/

The cases and examples presented in this chapter exemplify the limitations of traditional centralised infrastructures during emergencies, such as is the case of the invisible and isolated communities detected during the Haiti earthquake due to the lack of cellular network coverage. Another example is the assumption that Internet connectivity between victims, responders and bystanders is available and has sufficient quality, which can not be guaranteed in most emergency cases.

Consequently, this chapter presented Mobile Ad-hoc Networks (MANETs) as a technology aligned with the message from international humanitarian institutions and capable to empower people with a decentralised mobile communication through mobile devices.

The mobility these mobile devices afford, their capacity to build networks and their growing adoption in daily life, make these small scale computational resources accessible to a more ubiquitous digital world. For example, by 2015 mobile broadband subscriptions had an overall penetration of 12 times that of 2007.

Despite the effort invested in research on MANETs and MANET routing protocols, the proposed routing protocols have their limitations in real scenarios such as emergencies. Therefore, this chapter described the main features of MANETs, and also the main problems that MANETs are still facing. The following summary of limitations will appear again in Chapter 4, where the Information Centric Networking presented in Chapter 3 is applied in MANETs as a potential approach to address them.

- Routing to distribute data in MANETs is still an open problem, due to the unpredictable mobility models of mobile devices.

- Unlike wired networks, routing protocols in MANETs need to efficiently utilise mobile resources, particularly energy consumption.

- Real open source implementations for Android devices still remain few and most of them do not have up-to-date or well maintained repositories. They also require complex procedures such as rooting devices, which makes them unsuitable for a wider community of users.

Finally, this chapter listed available implementations for mobile devices, such as SPAN and BATMAN. Even these implementations utilise proactive routing protocols, they still are facing issues such as energy awareness and the assumption that mobile devices have already administrative privileges, which limit their practical usage.

CHAPTER THREE

INFORMATION CENTRIC NETWORKS

3.1 Introduction

The end-to-end nature of the Internet Protocol (IP), which allows the delivery of packets based on origin and destination IP addresses, is facing challenges due to the increasing demand in areas such as mobility, content distribution, and security [Pan et al., 2011]. Today's Internet user base has increased to 3.4 billion and Cisco Systems expects that by 2020 global traffic will grow by 22% per year. Much of this traffic involves the retrieval of content of some form. However, the Internet was never designed to service this kind of demand and the basic protocols do not support content retrieval but only end-to-end communication between named hosts. This gives rise to problems such as duplication of traffic as requests for popular contents go to the same server repeatedly [Hong et al., 2013]. However, Pan et. al (2011) highlight that TCP/IP's narrow waist makes it hard to modify the core of the Internet architecture. Consequently, efforts to efficiently organise content have been bolted on in the form of technologies such as web caches, content distribution networks and load balancers.

Compared with wired networks, MANETs based on TCP/IP are even less effective due to the high mobility of nodes, their decentralised nature, and the constant changes in the topology of the network. For example, in the case that a node moves from one subnet to another, the node needs to be assigned a new IP address within the new subnet. This is because IP addresses bind together the identity of a host and its location[1] [Atkinson et al., 2009]. In addition, IP-based routing approaches are characterised by a significant overhead in identifying and updating possible routes between nodes as discussed in the previous chapter.

[1] https://tools.ietf.org/html/rfc1498

Therefore, instead of the traditional host-to-host approach, it is desirable to consider an alternative, content-oriented communication model, *Information Centric Networking (ICN)* [Jacobson et al., 2009a, Tyson et al., 2012], which is discussed in section 3.2, and may represent a more viable solution to problems that IP-based MANETs are still facing. ICN aims to become an alternative architecture for the future Internet [Jacobson et al., 2009b, Trossen et al., 2010, Ahlgren et al., 2012]. The Named Data Networking (NDN) project[2] is working on the implementation of protocols for ICNs. The NDN design and software provide the starting point for the development of named-data networking in MANETs developed in this thesis. However, the design, implementations and evaluation of NDN is oriented to wired networks, and there is little work on NDN in decentralized mobile networks as well as on energy consumption.

This chapter presents ICN as an approach to support and enhance data distribution in MANETs. It aims to be part of the foundation to apply content-centric approaches in MANETs which are discussed in Chapter 4. This chapter starts by introducing the main features of ICN instances, and continues with comparing all relevant ICN instances based on their approach to forwarding, data retrieval, caching and mobility. The chapter concludes by highlighting attributes of NDN that makes it a suitable alternative to distributing data in MANETs.

3.2 About Information Centric Networks

The recently emerged research area in Information Centric Networking (ICN) aims to re-design the Internet to a content based architecture [Jacobson et al., 2009a, Ghodsi et al., 2011, Zhang et al., 2014, Xylomenos et al., 2014]. ICN is a promising candidate to replace the traditional host-oriented communication in wired networks. It does not depend on IP addresses as it shifts the Internet pair-wise communication towards a content centric model [Bari et al., 2012, Dannewitz et al., 2013, Ming et al., 2014].

In today's Internet architecture, popular content is frequently requested using origin and destination connections. Therefore, the completion of data exchange, which is dependent on the state of links between both ends, falls fully under the control of the responder. Thus, information providers are responsible for the infrastructure and resources to handle all individual requests. On the other hand, the content in ICN is tagged with unique names, and it is cached by intermediate nodes to optimise traffic when a request is satisfied from the information provider to the requester.

ICN secures the content rather than securing the connection between end nodes. In the ICN model nodes in the network are considered as dynamic storage since they are able to cache

[2]https://named-data.net

the content of a resource completely or partially. Therefore, the content is available for future requests made by other requesters. Nodes communicate with each other by requesting names of the content of interest instead of retrieving the content by sending IP packets to a destination, which is the mechanism associated with today's Internet communication [Jacobson et al., 2009b]. Additionally, ICN allows the fragmentation and distribution of content to intermediate nodes regardless of their locations. As the content is fragmented in the network, a requester or intermediates node can retrieve and cache it as a whole or in parts, even in the cases when connectivity between nodes is unpredictable or unstable [Meisel et al., 2010b].

3.2.1 ICN characteristics

This section introduces the fundamental features of ICN, caching, naming, forwarding, mobility and security. It also compares the ICN model to the traditional TCP/IP hourglass representation.

Caching Nodes in an ICN can store copies of the content they handle in a cache for a fixed or variable amount of time. Content can be cached by nodes that receive data from other nodes and can be shared with nodes outside the retrieval path. Caching is the most important aspect of ICN, as future requests for the same cached content can be satisfied by retrieving the content from the cache instead of requesting it from the origin. As a consequence, retrieving content from intermediate caches offers the benefit of overall reduction of network traffic [Ahlgren et al., 2012]. From the requesters perspective, caching reduces transmission delays and increases the possibility of faster retrieval of popular content [Zhang et al., 2013].

Naming Similarly to the hosts in today's Internet architecture, naming is a fundamental aspect of ICN [Ahlgren et al., 2012]. In ICN, the concept of naming refers to associating a unique identifier with the content rather than with the host. Names can be assigned to the full content, such as the name of a book in a PDF file, or to parts of the content, segments for short, such as a segment of a video file. The name of a content is independent of its location, application, storage or means of transportation. In ICN, each object can be requested by its name instead of its location as in the case of an IP-based architecture. The requester of the object must know the name of the content of interest [Ghodsi et al., 2011].

Forwarding Routing in IP-based communication is perhaps the most complicated function performed by a network. Nodes in the network are required to collaboratively exchange routing information, such as the status of the links, to discover the state of the current network topology [Papadopoulos et al., 2010].

In an ICN, a *consumer* requests content by sending an *interest packet* containing the name of the required content to neighbouring nodes. These can simply respond with the requested content if they find it in their cache. If they do not have the content, they need to make a decision as to how to forward the interest to other nodes that can either respond from their caches or forward the interest further in the direction of a *producer* that can provide it. The nodes in the ICN act as *forwarders* and the communication takes place in *store-and-forward* fashion. Whenever content matching the name is found, it is sent back using the *reverse path*, again using a store-and-forward approach. There are various approaches described in the ICN literature that follow this basic approach. A comparison between them is presented in section 3.3.

Mobility In contrast with the IP-based approach, ICN does not always retrieve content from the original source. Instead, the request for the content can be served by different intermediate nodes that have already cached the desired content. Consequently, ICN does not need to maintain connections between end nodes or between different network domains. It is also possible to have multiple producers provide the same content without having to employ mechanisms such as load balancing.

Mobility is intrinsically supported by ICN, and nodes do not need to be permanently connected to the network. They can leave and join at any time, and they can be located anywhere in the network. As a consequence, mobile devices do not require network re-configuration every time the device changes to a different network segment, which is an important contrast with today's Internet [Tyson et al., 2012, Tyson et al., 2013].

Security Since content in ICN is referred to by its name, the verification of the binding between the name and the content is essential to ensure that the receiver has received the correct and complete content (content integrity and authenticity). In content-centric communication, each packet comes with a digital signature that allows receivers to verify content authenticity [Ahlgren et al., 2012, Edens and Scott, 2017].

In ICN, attacks, such as denial-of-service, require different approaches compared to similar attacks in the traditional IP based communication. In ICN, the content is distributed across the nodes in the network and not in one specific node as it is the case of IP. Therefore, a denial-of-service cannot simply be launched by sending large numbers of request to the same data. Instead, it requires effort to stop services from more than one node of the network [Edens and Scott, 2017].

Compared to IP-based networks, ICN does not secure the host or end nodes to build trust. Instead, it puts more effort into securing the content. Just as security in an IP-based network relies on

a public key infrastructure (PKI) that authenticates hosts, security in a named-data network requires a PKI that authenticates producers of content.

Thin waist A final fundamental difference is that ICN suggests that the "thin waist" of the future Internet should be based on naming rather than host addresses of IP. Routing based on names allows users to focus on the content of the objects of their interest rather than the location of the host storing the desired content. In ICN, naming represents a common layer that supports upper layers, such as the application layer. It can be deployed on the top of lower layers, such as the link layer, and even can be layered on top of the traditional TCP/IP stack [Mahadevan, 2014]. For example, Figure 3.1 depicts a representation of the "thin waist" of the future Internet architecture according to an instance of ICN, Content Centric Networking (CCN).

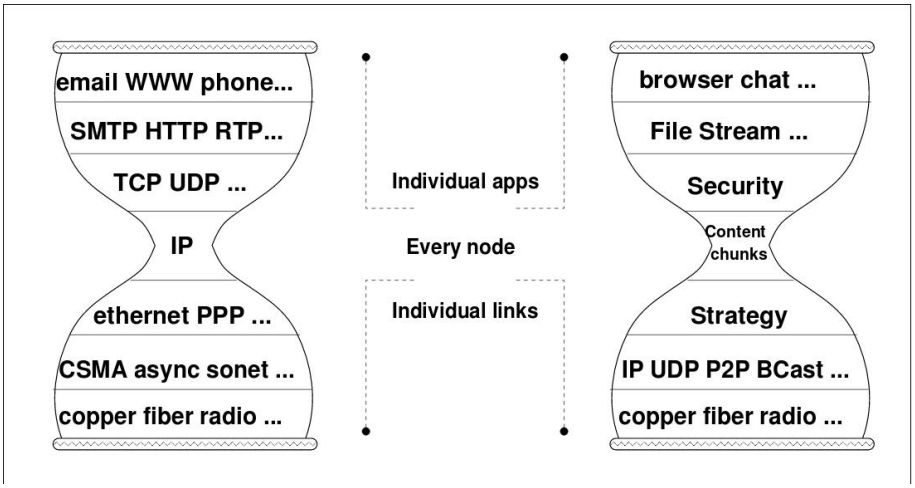


Figure 3.1: CCN "Thin waist" of future Internet architecture [Jacobson et al., 2009b]

3.3 Comparison of ICN instances

In the last few years numerous ICN instances have been proposed. Table 3.1 compares the different ICN instances available in public open source repositories. The comparison is based on the availability of academic literature such as the last published paper and the number of citations in Google Scholar[3]. Another criterion refers to the availability of up-to-date technical documentation to reproduce implementations of the ICN instances. This criterion includes whether a code is accessible, well maintained or documented.

Details about each instance are described in Section 3.3.1, which lists the description of each ICN instance based on four main features: forwarding, data retrieval, caching and mobility.

[3]https://scholar.google.co.uk/

Year	ICN instance	Cited Google Schollar	Last paper	Repo, Website, testbed	Mainteined	Funder
2007	DONA [Koponen et al., 2007]	1246	2007	NA	NA	UC Berkeley
2010	PURSUIT	NA	2011	https://github.com/fp7-pursuit/blackadder http://www.fp7-pursuit.eu/PursuitWeb/ http://www.psirp.org/index.html	NO (last event 2013)	FP7 EU project
2010	SAIL	NA	2013	http://www.sail-project.eu/ (Offline)	NA	EU Framework 7 Programme
2010	COMET	NA	2013	http://www.comet-project.org/	NA (last event 2013)	EU Framework 7 Programme
2010	CONVERGENCE	105 (as CONET)	2013	http://www.ict- convergence.eu	NA	EU Framework 7 Programme
2010	NDN [4]	2519	2017	http://named-data.net/ https://github.com/named-data	Yes	NSF's Future Internet Architecture Program
2010	Mobility First [Seskar et al., 2011]	60	2015	http://mobilityfirst.winlab. rutgers.edu/	NA	US Future Internet Architecture program
2009	CCN [Jacobson et al., 2009b]	2665	2016	http://blogs.parc.com/ccnx/	Less than NDN	Palo Alto Research Center,USA

Table 3.1: ICN instances comparison

As shown in Table3.1, the most cited projects are DONA, CCN and NDN, while the remaining approaches are less actively cited in the literature. As there are no recent publications about DONA, it has been excluded from the list of potential alternatives utilised in MANETs. Therefore, two currently researched and actively developed projects, Content Centric Networking (CCN)[5] and Named Data Networking (NDN)[6] are discussed in the next paragraphs.

CCN has been designed by Jacobson et. al and an implementation was deployed at Xerox's Palo Alto Research Center under the name of CCNx[7] [Jacobson et al., 2009b, Perino and Varvello, 2011, Mosko et al., 2015]. CCNx is being made available [8] and is distributed under the open BSD 2-clause License[9]. CCNx was pre-dominantly designed for wired networks, and currently no actively maintained and stable versions exist for wireless networks or

[5]https://wiki.fd.io/view/Cicn
[6]http://named-data.net/
[7]http://www.parc.com/,http://blogs.parc.com/ccnx/
[8]https://github.com/PARC/CCNx_Distillery
[9]https://opensource.org/licenses/BSD-2-Clause

for mobile platforms, such as Android, despite announcements made[10].

On the other hand, NDN[11], as a fork of CCNx, is also being made available as open source code[12] under the GPL 3.0 license[13]. The NDN team developed an NDN implementation, the NDN Forwarding Daemon (NFD)[14] for wired networks with the aim of it eventually becoming the core component of the NDN platform. NDN also deploys a wrapped version of NFD[15] for Android mobile devices. NDN for Android uses the Java Native Interface (JNI) to wrap and run NFD's C/C++ code. The source code of NFD is more actively maintained than that of NFD for Android, which may be attributed to the fact that NFD was originally designed for wired networks. Furthermore, please note that this simple wrapper does not provide any adaptations for NFD to run in ad hoc mode.

Comparing the available documentation of CCNx and NDN, NDN offers well maintained open source code with up-to-date releases, active technical support, and more extensive documentation including academic documentation, such as papers, technical reports, tutorials, simulators, and a testbed used by more than 20 universities.

The comparison made in this sections, which is presented in Table 3.1, shows that NDN is one of the most active ICN research and development projects, both in real and virtual infrastructures. Based on the comparison of NDN and CCN, NDN's NFD implementation offers more substantial academic and technical support. NDN's online developer community actively maintains the source code and aims to test NFD not only in wired networks, also in further areas, such as the Internet of Things and wireless networks.

3.3.1 Relevant ICN MANETs

Based on previous introduction, the following paragraphs briefly present relevant ICN MANETs approaches of Table 3.1. Each description is based on four criteria: forwarding, data distribution, caching and mobility. As a conclusion, one of the most relevant challenges is mobility of providers, for example maintain consistent routes between mobile nodes requires update potentially global locator information [Tyson et al., 2013].

[10] http://blogs.parc.com/2010/11/ccn-now-supports-android/
[11] http://named-data.net
[12] https://github.com/named-data
[13] https://github.com/named-data/NFD/blob/master/COPYING.md
[14] https://github.com/named-data/NFD
[15] https://github.com/named-data-mobile/NFD-android

Data Oriented Network Architecture (DONA)

- Forwarding: Each node, *Autonomous System (AS)*, has at least one *Resolution Handler (RH)*. Local producers send a REGISTER message to their local RH. Next, the RH propagates this registration to its parents or peering domain RHs based on locally established routing policies. Each intermediate RH stores the name of the object and the IP address of the RH forwarding the registration.

  Upon the consumers sending a FIND message to their local RH, the local RH replicates the message all the way up to the tier-1 RH, which peers with other RHs in other ASs. Therefore, the tier-1 RH is aware of all registrations in the entire network.

- Data retrieval: As soon as the FIND message reaches the producer, the data is sent back to the consumer using regular reverse path IP routing and forwarding.

- Caching: RHs can cache FIND messages for a specific period and send multicast updates in response to the consumers until they expire. In case a future FIND message requests the same object, the RH can supply the data directly from its local cache.

- Mobility: Mobile consumers can send FIND messages to their local and fixed RHs. However, messages from mobile producers need to register and unregister their REGISTER messages all the way to the Tier-1 RH, which causes a message overhead. In other words, consumers mobility is handled by changing RH hosts between networks [Tyson et al., 2013].

Publish Subscribe Internet Technology (PURSUIT)

- Forwarding: Paths between consumers and producers are calculated by the Topology Node (TM) in the Forwarder Nodes (FN). Path calculation in a dynamic topology may be subject to constant changes, which may intensively use the resources of mobile devices.

- Data retrieval: The approach is similar to that of forwarding, however it uses reverse path retrieval.

- Caching: content is cached by FNs, and it is available for future requests.

- Mobility: Consumers may send requests from any location, while producers needs to notify the TM about their new position. In other words, consumers mobility is handled by resubscribing the content being accessed [Tyson et al., 2013].

Scalable and Adaptive Internet Solution (SAIL)

- Forwarding: It is based on local and global Name Resolution System (NRS) nodes. Global NRS provides the location of the content and uses hop by hop forwarding (hybrid).
- Data retrieval: SAIL uses reverse path retrieval, which is the same approach used by NDN.
- Caching: The global NRS caches and centralises all most popular objects in the entire network.
- Mobility: SAIL uses the Late Name Binding (LNB) strategy in cases when nodes move.

Content Mediator architecture for Content aware Networks (COMET)

- Forwarding: It is based on local and global Content Resolution System (CRS) nodes, which have knowledge of the location of the data in the entire network.
- Data retrieval: The CRS provides paths to retrieve content.
- Caching: COMET uses a probabilistic caching scheme (ProbCache) and a Centrality scheme. ProbCache provides a means to determine the number of times an information packet should be cached on a path and assumes other CRSs have the same caching capacity.
- Mobility: Nodes can move and be detected by the exchange of location of the nearby CRS.

Convergence

- Forwarding: The forwarding approach of Convergence is hop-by-hop and by Border Nodes (BN). BNs provide a small routing cache. In case of insufficient routing information, BNs make requests to external Name Resolution Systems (NRS).
- Data retrieval: It uses reverse path data retrieval.
- Caching and Mobility: The approaches to both caching and mobility are the same as that of NDN, discussed next.

Named Data Networking (NDN)

- Forwarding: NDN utilises an internal *Forward Interest Base (FIB)* table to forward interest packets to the next node. By using an internal table, the Pending Interest Table

(PIT), which contains the list of interests that are not yet satisfied, echoes or loops in the network are eliminated.

Routing protocols, such as NLSR [Hoque et al., 2013], advertise prefix, as opposed to IP address ranges in OLSR. Additionally, NLSR could forward interest packets to the node with the best performance.

- Data retrieval: NDN uses the reverse path based on the crumbs left by the interest packets in the PIT table. In case the same content is received multiple times, all except the first are discarded automatically.
- Caching: The caching strategy of NDN can be described as on-path Content Store (CS) caching: the content is cached in the CS table of the nodes who forwarded the interest.
- Mobility: NDN offers the LFBL [Meisel et al., 2010b] protocol, which is described in section 3.4.7.2.

MobilityFirst

- Forwarding: It is based on Global Name Resolution Service (GNRS) nodes, which provide the full path between the Global Unique Identifier (GUID) of producers and consumers.
- Data retrieval: MobilityFirst uses GNRS.
- Caching: In this approach, each node has a local cache. In case the requested information is not stored in the local cache, the node requests the content from the GNRS.
- Mobility: Mobility is handled by a centralised GNRS.

CCN

- Forwarding: Similarly to NDN, forwarding in CCN is based on naming objects prior to distributing them across the network.
- Data retrieval: CCN utilises the same data retrieval approach as NDN.
- Caching: In CCN, objects can be obtained from a memory or from the wire.
- Mobility: Mobility in CCN is similar to NDN, due to NDN was inspired by CCN.

3.4 Named Data Networking (NDN)

The following section presents the main features of NDN, including the NDN platform architecture, NDN packet structure, and NDN forwarding strategies.

3.4.1 NDN architecture

The minimal basic configuration of NDN's architecture consists of consumers, forwarders and producers, as shown in Figure 3.3. Consumer nodes express their interest to retrieve some content from the network. Producer nodes generate or store the content of interest. Finally, forwarder nodes link consumers and producers by communicating the interest of consumers and relaying back content from producers. The following paragraphs describe the three roles of nodes in an NDN network in more depth.

Consumer Consumers are the nodes in a network that are interested to retrieve the content of a particular name. They express their interest by sending interest packets to nearby forwarders in order to receive the corresponding data packets. Consumers can disseminate their interest packets by using any supported transportation technology, using either unicasts or broadcasts. They send their requests to one or more nearby forwarders.

Once the consumer has sent the interest packet, there are two possible scenarios. The first one is when the consumer receives the desired content from a nearby forwarder, in which case the consumer declares that the interest to retrieve the content of a name was satisfied. In the second scenario, where the consumer does not receive any reply from neighbours, the consumer declares the request as unsatisfied and the decision to cancel or send a new request is left to the application. Additionally, the maximum time to wait before declaring the interest as unsatisfied is also left to the application.

Producer Producers create original content, assign a particular name to this content (see below). They notify one or more nearby forwarders of their existence through a registration process. Producers wrap the prefix of content they produce into a registration packet and send it to at least one forwarder, requesting prefix registration. The forwarder replies to these registration requests with registration acknowledgement packets, which contain the status of the registration process. After the registration process, producers will receive interest packets from consumers via forwarders and will respond to them by sending data packets containing matching content. If the requested content is too large for an individual packet, it is split into multiple segments as discussed in Section 3.4.6. Each data packet is signed to authenticate the producer and protect the data from accidental or malicious modification in the forwarder nodes or while in transit.

Forwarder Forwarders are the intermediate nodes that receive interest packets and redirect them to next hop forwarders or producers in order to find the requested content. Forwarders also use the reverse path to send back the content from the producer to the consumer. In an NDN network, forwarders play a similar role that routers play in IP-based networks. Unlike IP routers, however, they can cache content in their content store and use it to satisfy consumer interests directly rather than forwarding them in the direction of the producer. Most of the functionality of an NDN network is realised within the forwarders while the consumer and producer roles are kept relatively simple to aid application development.

3.4.1.1 Naming

Producers tag each content with a name, which is used by consumers when they need to retrieve any content. As depicted in Figure 3.2, the name of a content is formed by component names separated by a delimiter, which is the symbol ”/” in the case of NDN. The name of an object has a prefix that excludes the last *n* component names, where n is decided by each producer. Therefore, each producer can choose the name scheme according to their preference by defining the name space of an object or group of objects, which means to define a prefix and its structure using name components [Zhang et al., 2014]. An important feature of naming in NDN is that the length of the names are variable, which means that a name can be composed of a variable number of component names, where each component name can have a variable length.

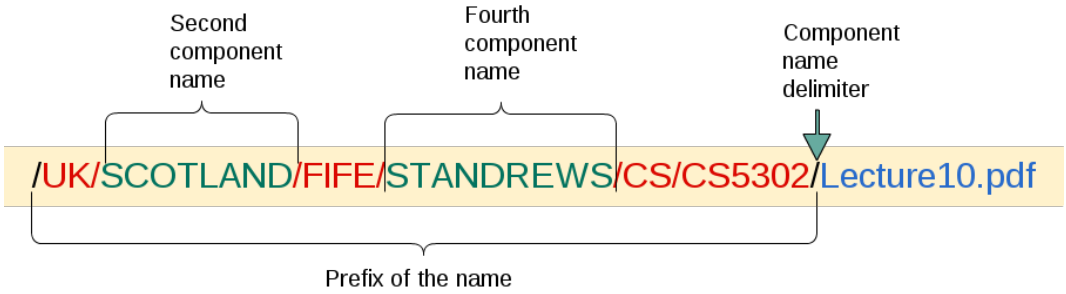


Figure 3.2: NDN name internal structure of an object

The communication in NDN is based on the exchange of two main types of packets: *interest* and *data*. Interest packets contain the name of the content interested to be retrieved, while data packets carry the content itself and its name. NDN reuses these two types of packets in other operations such as prefix registration and registration acknowledgement. To request a prefix registration in a forwarder, the producer wraps the prefix of the name in an interest packet and the result of the registration process is returned by the forwarder wrapped into a data packet, which NDN calls a registration acknowledgement.

Figure 3.3 depicts a consumer-producer scenario, where producers register the prefix of their own available content in nearby forwarders by sending a registration request. Thus, these nearby forwarders hold an updated map of prefixes associated with their respective producers that store the content. Consequently, in the case that a consumer sends an interest packet which name contains a prefix held by forwarders, the forwarders can simply redirect the interest packet to the corresponding producers. Then, producers can wrap the content into data packets and send them on the reverse path until the consumer is reached [Zhang et al., 2014].

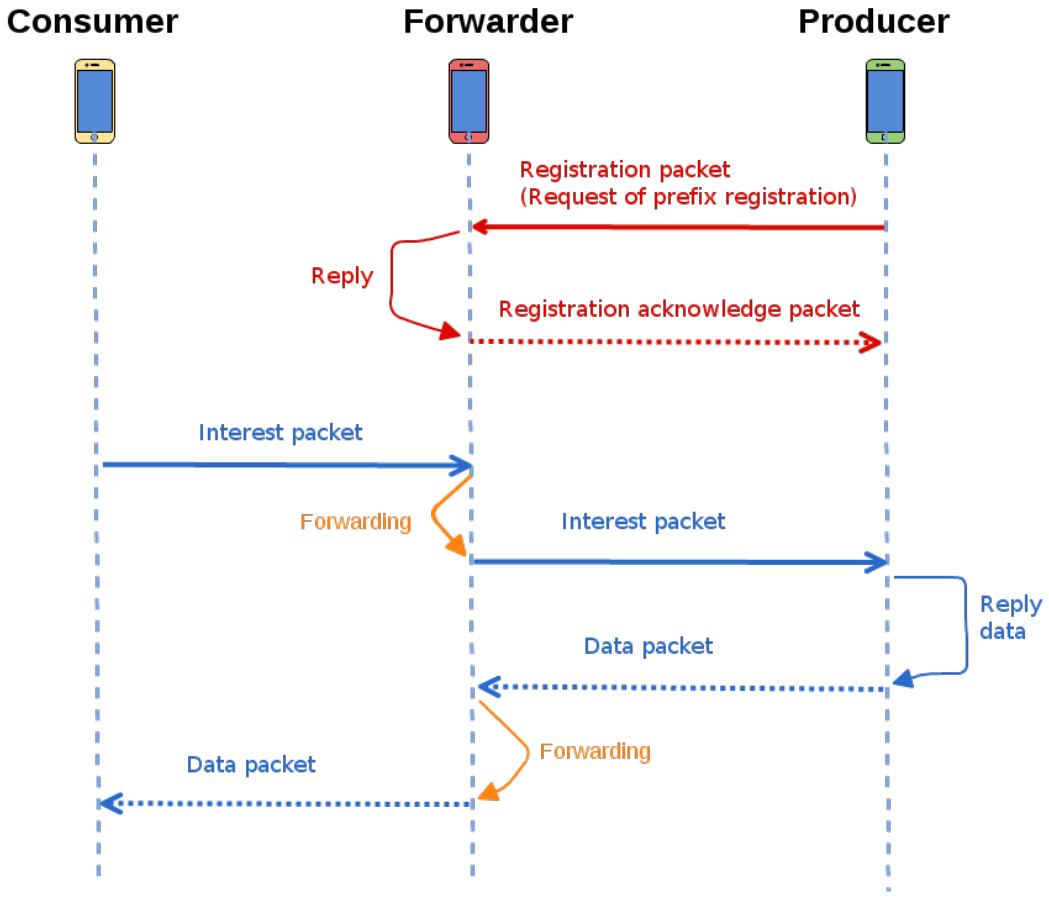


Figure 3.3: Example of a consumer-forwarder-producer scenario

NDN nodes send and receive packets through the so-called *faces*. Faces in NDN are what interfaces are for IP, providing physical or logical communication interfaces to other nodes or services. NFD nodes make forwarding decisions using using three internal basic tables: a Forwarding Information Base (FIB), a Pending Interest Table (PIT) and a Content Store Table (CS).

- **FIB:** NDN uses the FIB to forward interest packets. It holds a map of prefixes linked to faces towards potential forwarders or producers that store the content. This table is used

to decide which next face or faces the interest packet should be forwarded to. The main contents of the FIB include: next face, interest name, and timeout [Mosko et al., 2014a, Mosko et al., 2014b].

- **PIT:** PIT is used by forwarders to register names that have not yet been satisfied by a node, which means it represents a map of unsatisfied names associated to faces towards nodes who are interested in the content of the name. PIT plays an important role in forming the reverse path used to send data packets back to the consumer. Forwarders use the PIT to send data packets to the faces towards the nodes that expressed interest in them. Once the interest in the content of the name has been satisfied, the forwarder unregisters the name in the PIT [Mosko et al., 2014a, Mosko et al., 2014b].

- **CS:** The CS represents the local cache where a copy of the content is stored by the forwarder, CS represents a map of names and their respective content. The forwarder searches in the CS when an interest arrives. It forwards the interest packet only if the content of the name given in the interest cannot be found in the CS. Otherwise, the content is sent towards the consumer using the face from where the request originated [Mosko et al., 2014a, Mosko et al., 2014b].

Note that as shown in Figure 3.3, each time a forwarder redirects content of a name of interest towards the consumer, the forwarder caches the content and its name. By using caching, forwarders not only reduce traffic duplication in the network, it also improves performance on the consumer side [Wang et al., 2010].

3.4.2 NDN packet structure

As mentioned in Section 3.4.1, communication in NDN is driven by the exchange of interest and data packets. Figure 3.4 depicts the general internal structure of both packets.

At the time of writing, the available academic literature does not describe the packet structure in detail and through practical examples. Therefore, the following is based on the available technical description of the NDN 0.2-2 documentation[16]. Table 3.2 and Table 3.3 summarise the available definitions in the online documentation about the elements of an interest[17] and data[18] packet, respectively.

These descriptions contain only partial information of the format of NDN packets and do not explain in detail the interactions between the different roles. It has therefore been necessary to

[16]https://named-data.net/doc/ndn-tlv/
[17]http://named-data.net/doc/NDN-TLV/current/interest.html
[18]http://named-data.net/doc/NDN-TLV/current/data.html

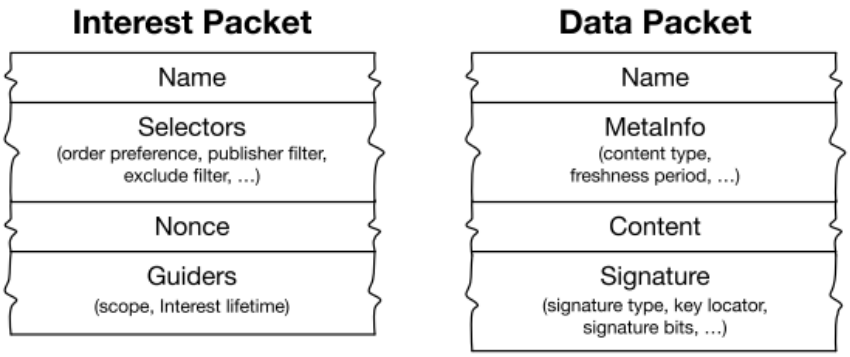


Figure 3.4: Interest and data packet structure [Zhang et al., 2014]

Field	Description
Name	Contains a sequence of component names.
Selector	Not available.
Nonce	This field helps to uniquely identify an interest packet, it is a randomly-generated long byte-string of 4-octet that combined with the name generates a unique identifier. It is used to detect looping Interests.
Guiders	Not available.

Table 3.2: Description of elements of the NDN interest packet

Field	Description
Name	Contains a sequence of component names.
MetaInfo	Not available.
Content	Not available.
Signature	It contains the description of the signature and the signature itself.

Table 3.3: Description of elements of the NDN data packet

reverse-engineer some of this information. One approach to observe more details of the internals of NDN packets is by capturing NDN packets through an NDN proxy[19]. Another alternative is to use generic network packet inspection with tools such as Wireshark[20] with NDN plugins[21]. The NDN proxy provides a more straightforward and customisable reverse engineering solution to trace NDN packet exchanges. Thus, a more detailed and complete description of the structure of NDN packets can be obtained by combining the available NDN packet specification[22] and the results of decoding of NDN packets captured by the NDN proxy.

3.4.3 NDN proxy

The NDN proxy provides a means to explore the structure of NDN packets in detail as well as their flow between nodes. NDN proxy is written in Java using the Java Client libraries for NDN[23] (JNDN), and needs to be placed between two nodes that communicate using the NDN protocol. To understand NDN as a communication protocol between nodes, it is necessary to dissect and describe the structure of interest and data packets. The NDN proxy facilitates this process by parsing the packets into Java objects, from which any required information can be extracted relatively easily using the JNDN API.

Figure 3.5 depicts an example of how the NDN proxy captures NDN traffic between a forwarder and a consumer, and between a forwarder and a producer. The flow is started by the producer, which sends a registration interest packet to the NDN proxy to announce its availability and the prefix of available content. The NDN proxy captures and copies locally this interest packet and then redirects it to the forwarder, thus the forwarder can register the prefix in its FIB table.

Once the registration is complete, the forwarder replies with an acknowledgement data packet to the NDN proxy, which makes again a local copy of the received packet and redirects it to the producer. Thus, the producer can receive the acknowledgement from the forwarder.

The second part of the flow starts when the consumer sends an interest packet to the NDN proxy. The NDN proxy makes a local copy of the interest packet and redirects it to the forwarder. The forwarder then checks whether the name of interest is in the CS. In this example, the name of the interest sent by the consumer matches the name cached in the CS of the forwarder, the forwarder replies to the NDN proxy with the corresponding content by sending a data packet. Finally, the NDN proxy receives the data packet and makes a local copy before redirecting it to the consumer.

[19] https://bitbucket.org/avoss/ndn-cd
[20] https://www.wireshark.org/
[21] https://gist.github.com/yoursunny/b7002336af5859737694
[22] https://named-data.net/doc/ndn-tlv/
[23] https://github.com/named-data/jndn

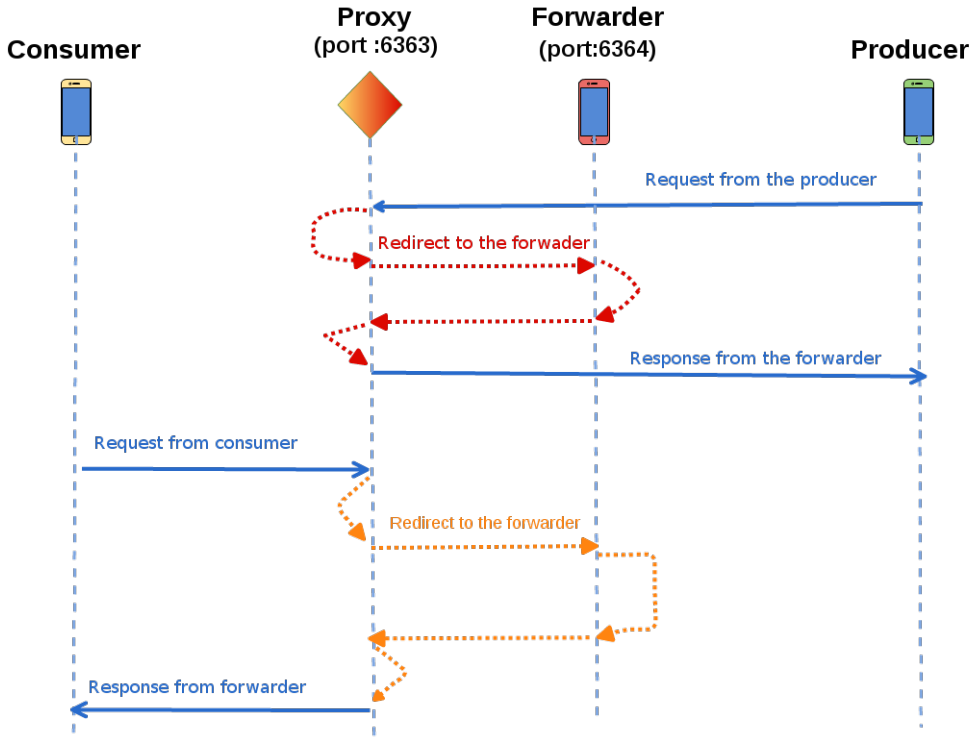


Figure 3.5: NDN proxy in front of a forwarder.

The flow of packets finishes once the consumer receives the data packet of the name of interest.

This example highlights that by placing NDN proxy in front of the forwarder in different scenarios, it is possible to capture all types of incoming and outgoing NDN packets to the forwarder. Therefore, a decoding of these captured packets can be used to analyse the internal structure of interest and data packets. The following subsections present the results of this byte by byte decoding process. Its analysis[24] has identified the following main types of packets: interest, data, registration, and registration acknowledgement packets. The byte to byte decoding process was completed by using the main type value table[25], and the control command table[26] available from the NDN specifications.

3.4.3.1 Interest packet

Figure 3.6 shows the internal structure of an interest packet, which contains the following fields.

- **Packet Type:** This is the first byte that appears in the packet and for an interest packet is 0x05.

[24] https://sites.google.com/site/meshawaremobilenetworking/ndn-packet-format
[25] http://named-data.net/doc/ndn-tlv/types.html
[26] https://redmine.named-data.net/projects/nfd/wiki/ControlCommand

- **Name Components:** This name field contains a list of name components. Each name component corresponds to a part of the name of the interest, and appears in the same order as they appear in the name. For example, the name /a/b/c has three name components, the first one is a, the second one is b and the third one is c, appearing in that order.

- **Selectors:** Selectors are optional elements that are placed after the name. Selectors are used to discover and select data that matches the interest name.

- **Nonce:** A nonce is represented by a random four bytes, which, combined with the name field, produce a unique identifier for the interest packet. The nonce is used to detect duplicated packets and to eliminate loops[27].

- **Interest Life Time:** This field represents the life span of the packet expressed in milliseconds. The default life time of an interest packet is 4000 milliseconds.

- **End of packet:** This is the last byte of the interest packet and it is coded with 0x00.

Note that the combination of name components and the nonce provides a unique identifier that is utilised to detect and avoid loops. Therefore, in the case that a forwarder receives two interest packets with the same name and same nonce, the latest one is discarded as it represents a loop. In the case that a content is divided into segments, and that the content has a version associated with it, the unique name associated with each segment has the following form: name + version id + segment id + nonce. The name, version id and segment id are carried as name components while the nonce utilises its own designated field.

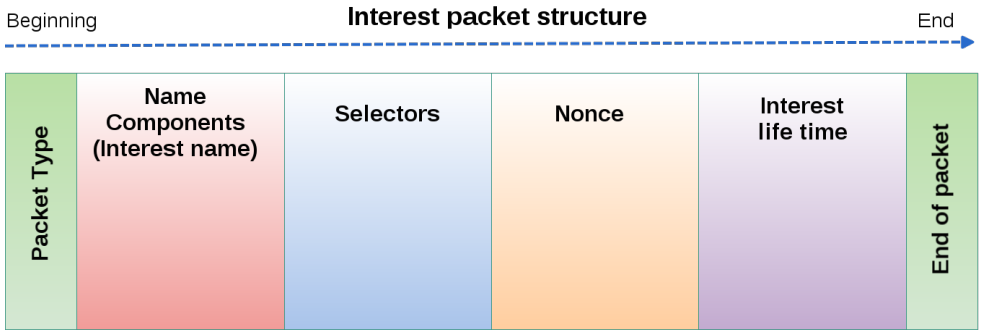


Figure 3.6: NDN interest packet structure.

[27] http://named-data.net/doc/ndn-tlv/interest.html

3.4.3.2 Data packet

Figure 3.7 depicts the internals of the structure of a data packet[28] composed of the following fields:

- **Data Packet Type and Name Component:** These two fields are similar to their corresponding fields in the interest packet. The data packet type for a data packet is 0x06.
- **Meta information:** This field provides extra information about the data packet, such as content type, time in milliseconds to mark a packet as stale, and the identifier of the final block of a sequence of segments.
- **Content:** This field stores the content that belongs to the name.
- **Signature Information:** This field contains signature information including the type of signature, an optional key locator and further signature-type specific fields.
- **Signature value:** The actual signature is stored in this field.
- **End of packet:** Similarly to the interest packet, this is the last byte of the data packet and it is also coded with 0x00.

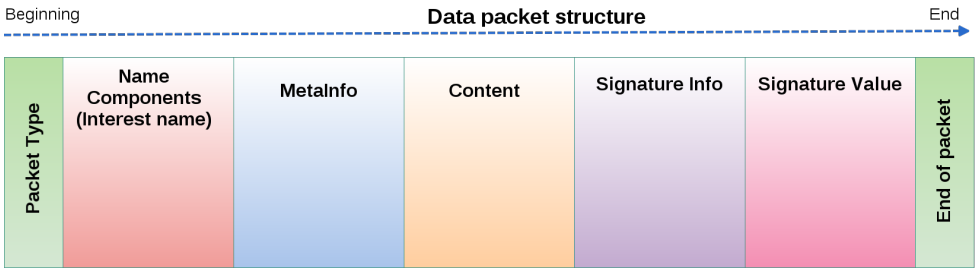


Figure 3.7: NDN data packet structure.

3.4.3.3 Registration packet

Registration packets are a specific type of interest packet that contain dedicated sub-fields wrapped in the name components. Registration packets are sent when a producer aims to register a prefix it handles with a forwarder. As shown in table 3.4, the sub-fields contain a dedicated registration name, the version and the segment identifier of the content, and the description and value of the signature.

[28] http://named-data.net/doc/ndn-tlv/data.html

Registration name	Description
/localhost/nfd/rib/register/	To request a registration to the local forwarder.
/localhop/nfd/rib/register/	To request a registration to a remote forwarder.

Table 3.4: Dedicated names for prefix registration request

Table 3.8 shows two specific registration names to register a prefix with a local and a remote forwarder. The difference between these two dedicated names is that a the name for local forwarder is to register the prefix in a forwarder located in the same node, while the name for the remote forwarder is to register the prefix in a different forwarder, not the local one.

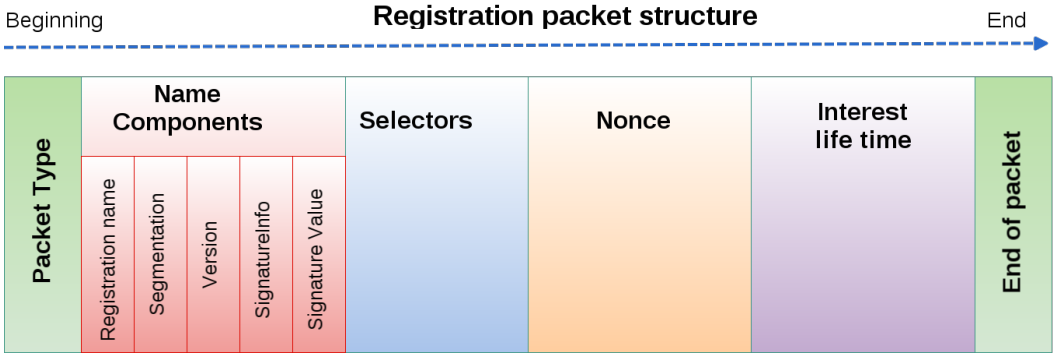


Figure 3.8: NDN registration packet structure.

3.4.3.4 Registration acknowledgement packet

The forwarder sends a registration acknowledgement packet, which is shown in Figure 3.9, to respond to the producer with the results of the registration process in the local FIB. Registration acknowledgement packets are data packets with associated control parameters, such as a status result code and status results text. The control parameters block may contain fields such as registration name, face identifier, uri, origin, cost, flags, strategy, expiration period and face persistence.

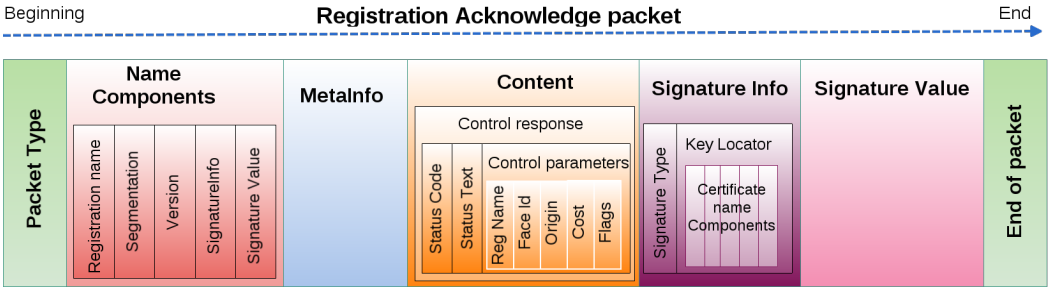


Figure 3.9: NDN registration acknowledge packet structure.

3.4.3.5 Negative acknowledgement packet

This data packet is utilised to inform to the requester about negative results of its request. Negative acknowledgement packets (NACK) are in fact data packets that carry additional messages such as unavailable route. They are usually utilised to inform that a request can not be satisfied at the moment, therefore the requester can take further decisions. The use of NACKS in ICNs is controversial. While they may solve some security problems such as interest flooding, they raise other problem themselves [Compagno et al., 2015].

3.4.4 Forwarding process

The forwarding process in NDN is depicted in Figure 3.10. This figure presents two cases. The first one is when an interest packet arrives at the forwarder, the forwarder verifies whether the content requested is present in the CS. If it is present, the content cached in the CS is send back to the consumer. If it is not present, the forwarder verifies whether it is present in the PIT. In the case that the name of the content is present in the PIT, the forwarder concludes that the interest was already requested and processed and only updates the PIT if the interface is new for the forwarder. In the case that the interest is new and it is not present in the PIT, the forwarder verifies if the FIB entries can satisfy the interest. In the case that the prefixes in the FIB can satisfy the name of interest, the forwarder redirects the interest to the next hop registered in the

FIB. Otherwise, the interest packet is drop or a NACK packet is sent to the consumer to notify that the forwarder cannot satisfy the request.

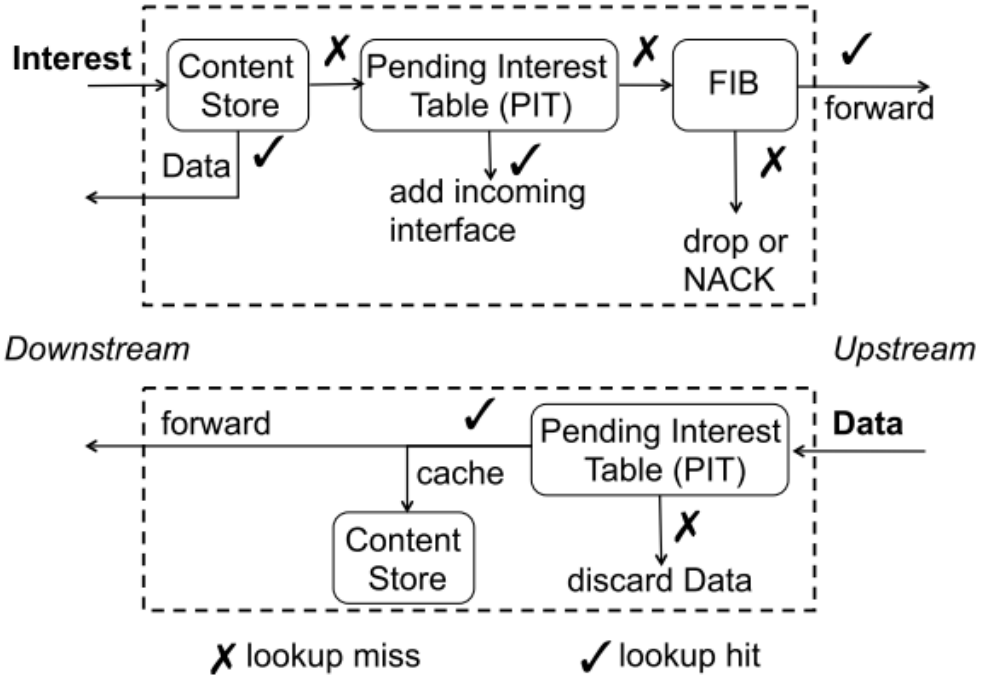


Figure 3.10: NDN forwarding process [Zhang et al., 2014]

The second case of Figure 3.10 corresponds to the case when a data packet arrives at the forwarder. Firstly, the forwarder checks whether the name of the content is present in PIT. If the interest is registered in the PIT, the data packet is sent to the interface that requested this interest. Otherwise, the forwarder discards the data packet but it may copy the content to the CS so it can be utilised to satisfy further requests for the same content.

3.4.5 Security

As introduced in Section 3.2.1, NDN content is distributed across the network, therefore intermediate nodes have a copy of the content. Security in NDN requires to securely retrieve the content regardless from where it comes from, rather than secure the destination host as in the IP-based approaches [Smetters and Jacobson, 2009].

In NDN, in the case an information provider receives a request of a specific content, it replies with NDN data packets that contain the requested content. In the case that the size of the content is large, the provider divides the content in segments, each with its own segment id, and signs each segment. Each of these segments are sent to the requester in the form of NDN data packets.

In addition to the signature, each data packet also carries the signing key name. To accomplish authentication, a trust model is required to validate signatures and keys. Note that in addition to being signed, there is nothing that keeps a provider to additionally encrypt the data to ensure confidentiality of the content. Just as it is not possible in a normal IP-based network to keep the sending parties anonymous, in NDN the identity of the content producer is not hidden. The NDN security model allows content segments to be located anywhere in the network and to be retrieved through any path [Afanasyev et al., 2016].

3.4.6 NDN Segmentation and versioning

At the time of writing there no academic literature focusing on segmentation and versioning. Therefore, this section refers to the NDN technical documentation available in the NDN project online site[29].

Segmentation Segmentation in NDN means dividing a large content into smaller pieces, where each individual piece is called a segment. Each segment is loaded into a data packet and is assigned a unique name. The name of a segment consists of the name of the content followed by a sequential number starting from zero. Figure 3.11 illustrates two characteristics of the segment number in a name. First, the segmentation is placed as a name component field. Second, segmentation is placed as the last name component of the name.

Figure 3.11: Structure of the name of a segment of the content “Lecture10.pdf”

Versioning NDN defines the version of a name as a name component field appended to the name of the content. The version of a content is a number that is set by the application, the bigger value of the version, the more recent the content. One simple way to implement this is to use timestamps as a value of the version. Figure 3.11 shows the location of the version name component of content ”Lecture10.pdf”.

3.4.7 Forwarding Strategies and Prefix Discovery

As any NDN node can have one or more faces to communicate with neighbours, *forwarding strategies* are used to decide to which of these faces a packet should be forwarded. These

[29] http://named-data.net/wp-content/uploads/2014/08/ndn-tr-22-ndn-memo-naming-conventions.pdf

strategies differ in the criteria of selecting the next face to forward an NDN packet to. NDN strategies make forwarding decision based on the information available in the FIB table [Yuan et al., 2012].

In order to make these decisions, the FIB needs to be populated with information, either through a manual process (similar to setting a gateway address in IP-based networking) or automatically through prefix discovery. Two approaches for FIB generation are discussed below.

3.4.7.1 Named-data Link State Routing Protocol (NLSR)

NLSR [Hoque et al., 2013, Lehman et al., 2016] is a proactive name-based routing protocol designed to work in wired routers such as in the example presented in the NLSR implementation [30]. NLSR was not designed for MANETs but since it is a relevant routing protocol for NDN, its general features are discussed in this section.

It utilises NDN interest and data packets to exchange routing information between routers. This exchange of information propagates link-states across the network and provides to routers multiple paths to each name prefix, which means to provide a mapping between prefixes and next faces. In other words, NLSR utilises NDN interest and data packets to populate the Forwarding Information Base table of NDN routers, and ranks the list of next hops to reach each name prefix based on the costs to reach the destination. In this way, a router can redirect an incoming interest packet to the next hop by applying a forwarding strategy using the FIB entries updated by NLSR [Hoque et al., 2013].

One important feature of NLSR is the utilisation of hierarchical naming to identify routers, routing processes, routing information, and keys. NLSR identifies each router by three components: the network it resides on (network), the specific site of the router (site), and an identifier of the router (identifier). The identifier component contains two subcomponents: the router tag and a label. For example, Table 3.5 describes the components of the name "/SONY/London/Pop1/%C1.Router/Router10".

Component name	Description
SONY	Name of the network
London/Pop1	Place where this router is located and its point of presence
%C1.Router	Router tag
Router10	Router label

Table 3.5: An example of naming of the router "/SONY/London/Pop1/%C1.Router/Router10" in NLSR

[30] http://named-data.net/doc/NLSR/current/ROUTER-CONFIG.html

NLSR disseminates network topology information through Link State Advertisements (LSA), which are used to advertise all prefixes of a router. NLSR utilises LSA to routing processes in order to establish and maintain routing information between neighbouring routers. NLSR disseminates two types of LSA: prefix LSA and adjacency LSA. Their respective format is depicted in Figure 3.12. The prefix LSA is utilised in the cases when a new prefix is added or removed and the adjacency LSA is utilised to detect failures or recovery any of the router's links. The general structure of the format of a LSA is: "/<network>/NLSR/LSA/<site>/<router>/<lsa-type>/<version>, where the name component "router" identifies the router who originates the LSA, the "lsa-type" is set to "name" for prefix LSA and set to "adjacency" for adjacency LSA, and finally the "version" component is a number incremented by one every time a router creates a new LSA.

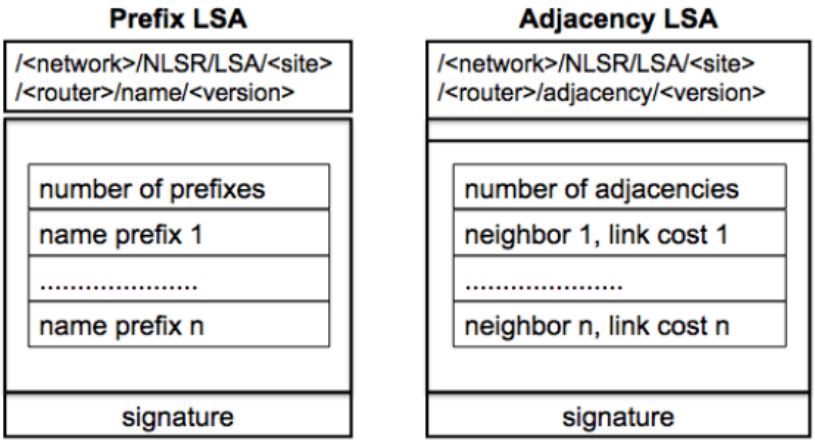


Figure 3.12: LSA types format for NLSR [Lehman et al., 2016]

NLSR routers also utilise Adjacency LSA to produce multiple next-hops applying Dijkstra's algorithm. This algorithm helps rank the next hops based in their costs to reach each destination. However, as the number of faces increases, computational costs rise, since the algorithm should be applied to all available faces to produce the cost of each possible path. This is an issue that remains to be solved [Lehman et al., 2016].

3.4.7.2 Listen First Broadcast Later protocol (LFBL)

Listen First Broadcast Later (LFBL) [Meisel et al., 2010b] is a receiver-centric forwarding protocol, that does not required pre-determined routing information from neighbours. Meisel et. al claim that LFBL has properties such tolerating physical mobility of nodes, and logical mobility of content, which makes it suitable in MANET scenarios. Physical mobility refers to the fact that the performance of LFBL does not depend on changes of the topology of the

network, and mobility of content means that any request of content can be satisfied by any node that can provide its content.

In this protocol, the nodes that receive a packet make their own decisions whether to forward the packet. In the case a node receives a packet, it pauses the forwarding and starts to observe the status of channel in the MAC layer. In case that the channel is not busy and no other node forward the packet, then the node forwards the packet itself. However, LFBL does not explicitly explain how to discover prefixes in the case that the FIB can not satisfy an interest. It implies that broadcasting the interests is the main mechanism to discover prefix in the network.

In this receiver-centric approach, the end-to-end communication is composed of two phases: the request phase and the data phase. In the request phase, requesters broadcast their interest about a particular content by disseminating the name of the interested content, and only responders who can satisfy these request reply with the respective content. By broadcasting requests, LFBL finds any available responder and distributes distance information that helps receivers to learn their distance from the requester.

In the request phase, the requester places the name of the content of interest in the field dataName, Table 3.13, and each forwarder updates its distance from the requester utilising the field srcDst. In the other end, once the responder receives the request, it produces a response packet, which LFBL header contains an update distance from the responder to the requester, field dstDist in Table 3.13. As a result, intermediate nodes that have forwarded response packets can redirect future request without broadcasting.

`seqnum`	A monotonically increasing sequence number assigned by the source node.
`acknum`	A cumulative acknowledgment number also set by the source node.
`srcId`	A unique endpoint identifier for the source node, generally its MAC address.
`dstId`	A unique endpoint identifier for the destination node.
`srcDist`	The distance between the source node and the most recent forwarder, modified at each hop.
`dstDist`	The distance between the most recent forwarder and the destination node, modified at each hop, if known.
`type`	The LFBL message type, which is either request (REQ), response (REP), acknowledgment (ACK), or a combined acknowledgment and request (ACK+REQ).
`dataName`	The name of the data being requested, provided, or acknowledged.

Figure 3.13: Fields of the LFBL header [Meisel et al., 2010b]

On the other hand, the data phase begins when the content starts arriving at the requesters, and it

is from this stage that responders and requesters continue to exchange requests and content until requests are ceased or responses are not longer received.

In LFBL, receivers make forwarding decision based on two criteria: whether it is an eligible forwarder and how long it should wait before forwarding the packet. The first criteria refers to the decision whether the receiver is closer to the destination than the previous hop from where the request has received. To answer this criteria, the receiver compares the dstDist information in the LFBL header of the received packet with its local buffered distance value, Table 3.13.

The second criteria refers to the time to wait before forwarding the request, which is called the listening period. By choosing an amount of time to wait, forwarders can avoid collision and prioritisation. Collision avoidance means to reduce the possibility that two eligible forwarders wait for the same listening period, transmit simultaneously and cause a collision. To reduce this possibility, the state of channel in the MAC layer should be exposed to allow nodes to transmit only when the channel is clear. In the other hand, prioritisation means to choose the best next hop based on the distance to the responder, which implies that nodes closer to the producer are eligible to forward the request.

One of the disadvantage of LFBL is the case where the distance metric between the receiver and the responder is overly optimistic. In this case, many receivers assume that they are eligible to be a forwarder, which causes extra overhead and unnecessary contention for the channel. In the other hand, if the distance is pessimistic, receivers assume they are not eligible forwarders, therefore they stop forwarding and packets can be lost.

3.4.7.3 Broadcast-based self-learning

Broadcast self-learning [Shi et al., 2017] builds the FIB by broadcasting the first interest packet and it populates the FIB by observing the return path of the corresponding data packet (reverse path). Therefore, future interest packets are forwarded via unicast using the FIB entry. Figure 3.14 illustrates that node C broadcasts the first interest until reaching B. When B sends the data packet, nodes S and R build their own FIB, which allows future interest with the same prefix to be forwarded using unicast instead of broadcast.

A practical example is an application that loads multiple content objects with names having a common prefix. The first interest packet is broadcast in order to add a corresponding entry in all the FIBs of intermediate forwarders. After this, the rest of the interest packets just require unicast as the forwarders are already aware of that prefix. However, a broadcast is required again if a the prefix is different in subsequent interests.

Additionally, once the FIBs entries are built, the further interest with the same prefix can be

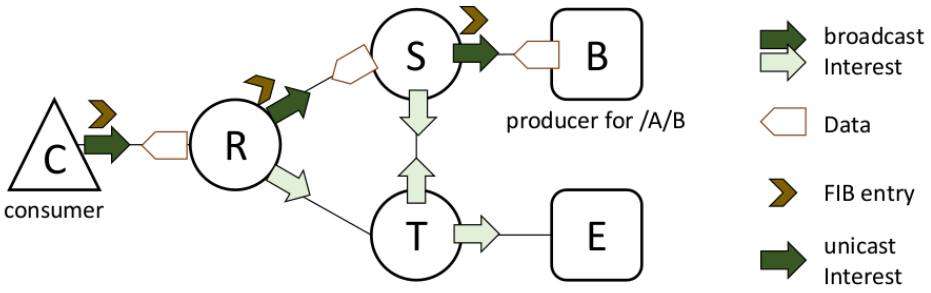


Figure 3.14: Broadcast-based self learning in NDN [Shi et al., 2017]

satisfied only if the route to the producer remains unchanged, such as in most of the cases in wired networks. However, this cannot be guaranteed in MANETs due to the mobility of nodes and the constant changes in the network topology.

3.4.7.4 Reactive Optimistic Name based Routing (RONR)

RONR is similar to the broadcast-based self-learning approach described above, with the difference that RONR was applied in an Internet of Things environment. In the case that a FIB is empty or no FIB entries match the prefix of the name of the desired content, RONR floods only one single initial interest. Therefore, RONR configures the FIB by using the information from the reverse path taken by the data packet towards the requester [Baccelli et al., 2014].

Some of the shortcomings include that RONR assumes the entire content is stored in one single node, which is reasonable in IoT. However, it is not necessarily the case in MANETs. Additionally, in RONR, availability of a path to the publishers is desired, otherwise, the subscriber has to flood again to discover a new route to an available publisher. In the case of MANET, due to nodes mobility, MANETs cannot guarantee constant availability of route paths towards the publisher, as availability and connectivity of nodes to the network may change at any unpredictable time.

3.4.7.5 Neighbourhood-aware Interest Forwarding (NAIF)

Forwarding in NAIF has based on the periodical evaluation of forwarding statistics, which are collected by each forwarder. Forwarding statistics in NAIF include the estimation of two rates. The first one is the data retrieval rate, which is the ratio of the number of successfully retrieved data packets to the number of interest packets sent by the forwarder. In the case that the data retrieval is low the incoming interest packet is discarded locally. The second one is the forwarding rate, which is the fraction of incoming interest packets that a forwarder redirects

[Yu et al., 2013]. NAIF does not explain clearly how the FIB entries are configured when the incoming prefix does not match or when the FIB is clear at booting time.

3.4.7.6 Best Route Strategy

This strategy forwards the name of interest to one next hop with lowest routing cost. In the case that the interest is not satisfied, then the interest is forwarded to another unused lowest-cost next hop. This also means that the consumer requires to re-send the interest multiple times in order to retrieve the correct next hop. The concept of low cost is not defined in the technical report[31] and the source code only validates whether the next hop violates the scope (local or non-local), does not forward back to the same face and that the next hop is not used[32].

To improve performance, this strategy informs the consumer with a Nack packet in the case that there is not eligible next hop in the FIB. Therefore, consumers can decide to retransmit, which may cause further delays. Additionally, repeated re-transitions increase the bandwidth overhead towards the consumer[33].

3.4.7.7 Multicast strategy

Multicast strategy forwards the interest to all next hops available in the FIB for a particular prefix match. This strategy requires an FIB to have all necessary entries in order to be able to forward every interest. This strategy does not provide a mechanism to discover prefixes, consequently, it is not suitable for this purpose[34].

3.4.7.8 Client Control Strategy

This strategy enables consumer applications to choose which outgoing face to forward each interest. In the case that the face is unavailable, the interest is dropped[35]. The source code of this strategy does not configure the FIB, therefore it is not suitable for this purpose[36].

3.4.7.9 NCC

This is a reimplementation of the CCNx 0.7.2 default strategy and does not include a mechanism to configure FIB, as the entries for this table are manually updated. The name NCC is just the

[31] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 42
[32] https://github.com/named-data/NFD/blob/master/daemon/fw/best-route-strategy2.cpp line 73-94
[33] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 42
[34] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 43
[35] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/ page 43
[36] https://github.com/named-data/NFD/blob/master/daemon/fw/control-strategy.cpp, line 35-50

reverse name of CCN, it has not any particular meaning[37].

3.4.7.10 Access Router Strategy

This strategy fits cases where the producer is one hop away from a router and single-homed. This strategy is similar to the Broadcast based self-learning strategy. It populates the FIB by learning from the reverse path when the data packet is retrieved to the consumer[38].

3.4.7.11 Auto prefix propagation

The auto prefix propagation registers prefixes to a single connected gateway router or one next hop. This NDN mechanism does not configure the FIB of farther nodes, nodes which distance from the router is more than one hop. It was designed mainly for wire networks, and it can not be applied directly in MANETs to configure intermediate nodes[39].

3.5 NDN and MANETs

NDN, introduced in detail in Section 3.4, appears to provide a viable approach to make MANETs more effective and efficient [Meisel et al., 2010a]. Thus, in the rest of this chapter, the application of NDN in MANETs is presented. Following an introduction to utilising NDN in MANETs, the challenges of deploying NDN in a MANET are described. Finally, other alternatives, such as CDN and ILNP are discussed in Section 3.6 and 3.7, respectively.

Table 3.6 summarises the main ICN instances and their suitability for MANETs. From this table, CCN and NDN emerged as candidates but as NDN offers more technical support and academic literature than CCN, NDN was selected as particularly suitable for MANETs.

In contrast with IP based host-to-host routing protocols, by applying NDN in MANETs, mobile nodes are not required to spend resources on calculating paths between end points [Meisel et al., 2010a], and content retrieval requires to know the name of the content of interest rather than the IP addresses of the node who stores it. In comparison with host-to-host communication, in NDN content is on nodes. Content does not necessarily need to be retrieved from origin nodes, content can also be retrieved from intermediates nodes that previously cached the same content. These and more details are discussed in Section 4.2.

As opposed to traditional IP-based networks, in which each object is retrieved entirely from the same information provider node, NDN offers distributed caching: each cached object can be

[37] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 43
[38] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 43
[39] https://named-data.net/publications/techreports/ndn-0021-6-nfd-developer-guide/, page 56

ICN instance	Is it suitable for MANETs in emergency scenarios?
DONA	No. DONA centralises all registrations in a local RH and Tier-1 RHs, In emergencies, local and Tier-1 RHs can not be guarantee to be available.
PURSUIT	No. This is due to PURSUIT has high dependency of the network on one node, which is the TM. MANETs, in emergencies, can not guarantee that the TM node will be present and available all the time.
SAIL	No. SAIL caching approach is centralized, A Global NRS is required to be available as this node provides the location of the content. The availability of this node can not be guarantee in emergencies scenarios.
COMET	No.Similarly to SAIL, COMET stores data location in one central node, the CRS.
CONVERGENCE	No. CONVERGENCE requires constant availability of an external, the NRS. Similarly, the nature of an emergency scenario can not guarantee the availability of an NRS.
NDN	**Yes**. NDN is not dependent on dedicated or centralised nodes. Instead, each node independently make decision to forward requests and retrieve content. This approach does not depend on traditional centralised infrastructure.
Mobility First	No. Mobility first is a combination of host and name based communication, which makes not suitable in cases of emergency scenarios.
CCN	**Yes**. Similarly to NDN, CCN is independent of the topology and the data transmission infrastructure. However CCN is less maintained and documented than NDN.

Table 3.6: ICN instances and brief of their suitability for MANETs in emergencies

fragmented and located in more than one caching node and in one or more intermediate nodes. This feature is highly desired in MANETs due to the mobility of devices, for example in cases such as retrieving fragments of videos from different intermediate nodes at same or different times.

Another advantage of using NDN over the traditional IP-based protocol in MANETs is that NDN provides an abstraction over the transportation type between nodes. Therefore, NDN can be perfectly functional on top of the link layer or above, as it does not need to be aware of the state of the connection, their reliability, or order. In NDN, packets are independently named and they are not part of any conversation between nodes.

In conclusion, based on the key features of ICN and a comparison of available ICN instances discussed in Section 3.3, NDN appears to be the most applicable in MANETs. Thus, NDN has been adopted as the foundation of this thesis.

3.6 Content Distribution Network (CDN) and NDN

Content Delivery Networks or Content Distribution Networks (CDN) are a set of dedicated data centres, technically called Point of Presence (PoP) that are distributed strategically across the world. CDN services are contracted by content publishers, such as Facebook, Netflix, or CNN to host and distribute their content. Therefore, user requests to retrieve content from publishers

are forwarded to the nearest PoP through DNS redirection, HTTP redirection, or URL rewriting [Shafiq et al., 2014]. Origin servers of the content publisher are only contacted by the PoP when the PoP has not cached the requested content. In this case, the origin server sends the content to the user through the PoP. Thus, PoPs are able to cache the content and utilise it for further requests. Content publishers not using CDN services are likely to face load and performance challenges in case their content becomes popular [Shafiq et al., 2014, Saroiu et al., 2002].

CDNs offer faster content delivery, higher content availability, and less traffic to origin server publishers. From a security perspective, CDNs offer better protection against malicious attacks, such as Denial of Service by replicating content in different locations, so blocking access to specific content becomes more difficult [Sivasubramanian et al., 2004].

Akamai Akamai [Nygren et al., 2010] is one of the most popular and largest CDNs consisting of tens of thousands of servers that distribute different content types such as static web sites, streaming media, or dynamic applications. Akamai was designed with the following main components.

- Mapping system. The domain name of a URL address typed in a browser is mapped to an IP address of an edge server close to the user. Each edge server is part of a large collection of servers located in sites around the world.

- Transport system. This component transports content between the origin servers and the edge servers. Hence, edge servers can respond to users by using internal cached content and using the content carried by the transport system. A typical example includes when users browse dynamic pages on a web site.

- Communication and control system. It is used to communicate control messages, status information, and updates in the configuration.

- Data collection and analysis. The data collection and analysis component is used for logging, alerts, billing and analytics.

- Management portal. This component serves two main purposes: configuration management and user interaction visibility.

NDN versus CDN CDN and NDN are both content-based data distribution networks. However, there are a number of differences between them. Content providers pay to store copies of their popular content on CDN regional data centres, while by utilising NDN, each forwarder can have

a partial or full copy of the content, in which case, the entire network can behave as one big content delivery network [Edens and Scott, 2017].

Another difference is that NDN is a protocol that can run on top of any data layer technology or layers above, and sends packets across the network using best effort packet delivery. In contrast, CDNs are represented by a large overlay infrastructure, with an expensive service, and specific to contracted applications. NDN overcomes these limitations and democratises content distribution.

CDNs offer scalability in case the same content is requested by different users, and it provides a good example of an implementation that overlays on today's TCP/IP architecture. However, in comparison to NDN and in cases when the network traffic is heavy, NDN offers a higher Quality of Service (QoS) due to it having higher concurrency and lower loss and latency [Ma et al., 2014]. Another difference between CDN and NDN is that, under the same topology and cache size, NDN performs better than CDN. This is because NDN offers an anycast mechanism, which improves availability and performance [Ma et al., 2014].

An additional difference is that a CDN runs services at the application layer, while NDN runs directly at the network layer to redirect packets along the best path. Furthermore, in a CDN, requests from users need to be redirected into the CDN system. This limitation can usually be solved by providing a DNS service for the domain that is served [Ma et al., 2014]. In NDN, dissemination removes the need of an external DNS and at the same time eliminates all requests that are identical.

As illustrated by nCDN, which embeds NDN in an existing CDN, NDN can simplify the design of a CDN, and increase robustness and enhance security by signature verification [Jiang and Bi, 2014]. While in a traditional CDN, requests from users are mapped on to the nearest PoP [Ma et al., 2014]; in nCDN, requests for a content are directly forwarded by the node who has cached it. Compared to traditional CDNs, nCDN offers better reliability and scalability, and can handle heavier load pressure, in particular in a dynamic network state [Jiang and Bi, 2014].

3.7 The Identifier Locator Network Protocol

The Identifier Locator Network Protocol (ILNP) is another attempt to address the limitations of today's Internet architecture, in particular, to introduce multi-homing [Atkinson and Bhatti, 2012]. ILNP is another promise of a "new generation of Internet". ILNP replaces IPV6 addresses with a non topological name space called the *Identifier*, and a topological name space called the *Locator*. ILNP addresses can be built from IPV6 addresses by splitting the 128 bits to two 64 bit parts,

where the lower part is assigned to the Locator, while the remaining 64 bits are assigned to the Identifier. The following paragraphs present the main characteristics of ILNP, and a discussion of utilising ILNP name spaces to allow integration with MANETs.

Locator As shown in Figure 3.15 [40], the Locator is represented by the first 64 bits of the IPV6 address and it is tied to the topology of the network. Locators are not used to establish and maintain sessions in upper layers, such as the transport or application layer, they are used for routing purposes only. The value of Locators changes with network topology changes, and these updates are communicated to the network to keep current sessions active.

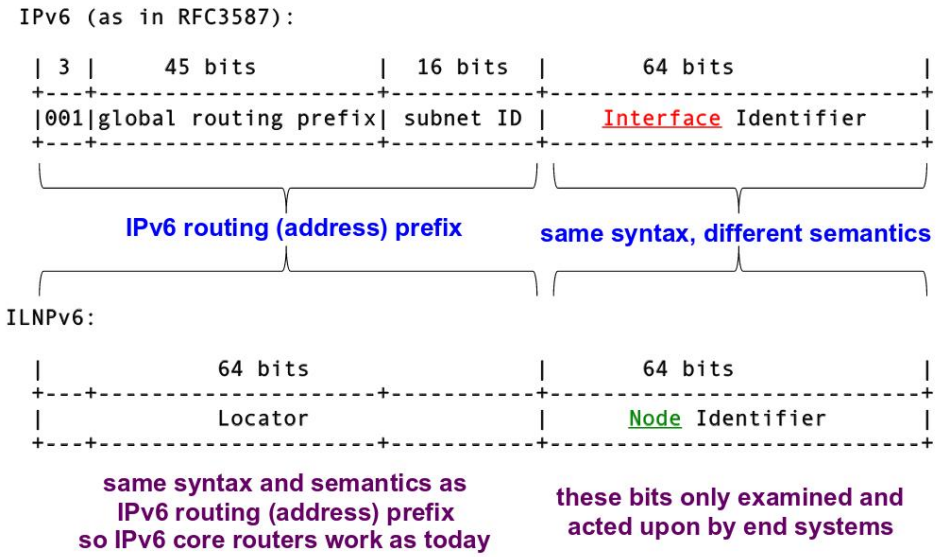


Figure 3.15: ILNP packet versus IP packet.

Identifier The Identifier, represented by the 64 bits following the Locator, is used to uniquely identify nodes, and not for packet forwarding. It is visible for upper layers and is not binding to an interface or Sub-network Point Attachment (SNPA) [Atkinson, 2005]. The uniqueness of the Identifier is ensured by using the MAC address of some hardware of the ILNP node.

Combining the Locator and the Identifier, ILNP offers services not supported by IPV4 and IPV6, such as mobility and multi-homing. In case a node moves between networks, only the Locator is updated on each network and the Identifier remains the same across all networks and throughout the life of a given transport layer session. Transport layer protocols that include the ILNP Identifier in their transport state, are not aware of changes of the Locator in their sessions. Consequently, sessions are not affected when the node moves from one network to another. Therefore, the location of the node does not impact the transport layer sessions between nodes.

[40] https://saleem.host.cs.st-andrews.ac.uk/talks/2010/ilnp_nanog50/2010-10-03-ilnp_nanog50-v4_final.pdf

Using ILNP does not necessitate rewriting applications in the application layer. Therefore, existing IP-based applications should be able to work over ILNP. However, using the Identifier within application layer protocols is not recommended.

Being a host-based communication protocol, compared to IP packets where the source and destination address are required, ILNP also requires a pair of Locator-Identifier of the source and destination host. At the transport layer, a pre-requisite for the IP model is to know the source and destination IP addresses and port numbers in advance. Conversely, ILNP requires a source and destination Identifier, a transport protocol, and source and destination ports. These differences are depicted in Figure 3.16.

Protocol layer	ILNPv6	IP (v4 and v6)
Application	FQDN	FQDN, IP address
Transport	Identifier, *ID* (+ port)	IP address (+ port)
Network	Locator, *L*64	IP address
Link	MAC address	MAC address

Figure 3.16: Use of names in ILNP versus IP [Atkinson and Bhatti, 2012].

ILNP integration with MANETs The main benefit of ILNP in MANETs is the inter-MANET communication and communication with non-MANET nodes by utilising the naming approach of ILNP, which is depicted in Figure 3.16. As in ILNP mobility and multi-homing are supported, by using identities to name nodes and values of locator, for example network prefixes, to name networks, ILNP allows integration with MANETs [Bhatti et al., 2011, Atkinson et al., 2009].

In a mobile network scenario, the *Locator* of a mobile node changes when the mobile devices changes its location, however the value of the Identifier keeps constant along all sub-networks. For example, in the case that a mobile devices moves to another IP based sub-network, the value of the Locator will change by a suitable local value and provided by router advertisements. ILNP allows to hold more than one Locator values concurrently, which facilitates multi-homing. Similarly to multi-homing, node mobility and network mobility follows essentially the same mechanism.

ILNP does not have its own approach for cases such as neighbour discovery of mobile devices (identity discovery) but instead utilises existing IPv6 mechanism such as SEcure Neighbor Discovery (SEND [41]). SEND utilises the IPv6 Neighbour Discovery Protocol (NDP)[42], which

[41] https://tools.ietf.org/html/rfc3971
[42] https://tools.ietf.org/html/rfc2461

is utilised by both nodes and routers for Neighbor Discovery (ND), Router Discovery (RD), Address Autoconfiguration, Address Resolution, Neighbor Unreachability Detection (NUD), Duplicate Address Detection (DAD), and Redirection. In addition, no implementations of ILNP were found that were tested on platforms such as Android mobile devices or inMANETs, which keeps open these areas as part of future research.

3.8 Chapter summary

This chapter introduced Information Centric Networking as an alternative future Internet architecture. ICN is a content centric approach that distributes data based on the name of the content instead of based on the addresses of the origin and destination nodes. In comparison to IP-based approaches, ICN secures the content rather than the connection between end points. Considering a consumer-producer scenario, ICN tags each content with a unique name, which is disseminated by the consumer into the network until reaching the information provider, i.e. the producer. Then, the producer replies with the content matching the name to the consumer utilising the reverse path. The path is built by the intermediate nodes or forwarders when the interest is forwarded. Additionally, this chapter has described the main elements in ICN design such as caching, naming, forwarding, mobility, and security.

A comparison of popular ICN instances is presented in combination with the details of each instance. The results of the comparison show that NDN is one of the most active ICN research and development projects both in real and virtual infrastructures. Therefore, Section 3.4 outlines NDN architecture, NDN packets structure and their interactions. Additionally, actual forwarding strategies in NDN, segmentation and versioning are also discussed in Section 3.4.

Even though NDN has an extensive academic and technical documentation, there is still little work done on real MANETs. In particular, there are no evidence of cases, such as how NDN behaves on MANETs using real mobile devices, such as Android. Also, at the time of writing, no literature was found regarding efficient use of energy from batteries when MANET devices exchange NDN packets and how they perform compared with traditional IP-based approaches, topics that are covered in Chapter 4.

Finally, the last sections of this chapter discuss and compare NDN with other technologies such as Content Distribution Networks (CDN) and ILNP, and also provide a summary of one of the most popular CDNs, Akamai. This chapter in combination with Chapter 2 provides the literature review that motivates and informs the development of nMANET, which is discussed in Chapter 4.

CHAPTER FOUR

NAME-BASED MOBILE AD-HOC DATA NETWORK (NMANET)

4.1 Introduction

Based on the discussions presented in Chapter 2 and Chapter 3, the aim of this chapter is to discuss the application of NDN in MANETs and the nMANET environment proposed to address reproducibility of tests and experiments. This chapter is organised as follows. It starts with a summary of the challenges of MANETs that limits reproducibility. Next, the proposed approach of this work, nMANET, and its main characteristics, and requirements are presented. These requirements provide the foundation for the design, implementation and testing of the Java NDN Forwarder Daemon (JNFD), a prototype implementation of nMANET, which is detailed in Chapter 5.

4.2 The nMANET approach

The motivation for using NDN in MANETs it to address the traditional problems of MANET implementations based on IP as well as to address the shortcomings of TCP as a protocol for content distribution. This section presents a summary of the limitations of traditional MANETs that limits reproducibility, followed by a discussion of observations from the academic literature on how NDN features can potentially address these. Finally, the section concludes with describing further benefits of NDN identified in this work that provide the groundwork and motivation to create nMANET.

4.2.1 Challenges in MANETs that limit reproducibility

Table 4.1 shows a summary of the main limitations and challenges MANETs are still facing after two decades of development (for an early review of work see [Royer and Toh, 1999]). These limitations are discussed in Chapter 2 and are the main motivation to create the nMANET approach in order to address these shortcomings and ensure experimental reproducibility.

Topic	Description
Routing	To efficiently distribute data in MANETs is an open problem due to the unpredictable mobility of mobile devices.
Mobile constraints	Unlike wired networks, routing protocols in MANETs are required to utilise limited resources efficiently, such as the energy remaining in batteries.
Lack of open source implementations	Open source implementations for mobile devices remain limited. Additionally, existing implementations are not up-to-date or well maintained and require relevant technical skills, which is likely to impede their adoption outside computer science research.

Table 4.1: Main MANETs limitations and challenges

Routing As per Section 2.5.2, the mobility of mobile devices in a MANET makes unpredictable changes in the topology of the network over time. Therefore, in the cases such as data distribution, MANETs require a protocol to frequently discover new paths between endpoints and update routing information of intermediate nodes to reflect the changes in the topology.

Consequently, routing in MANETs is still an open problem, and requires to be approached to ensure efficient utilisation of mobile constraints such as energy consumption from batteries. Additionally, it is important to contrast the current IP based routing approaches against new ones such as NDN. Analysis of the benefits and shortcomings of applying NDN in MANETs can lead researchers to investigate and propose new features such as the ones proposed by nMANET.

Mobile constraints Mobile constraints have influence in the life time of a MANET, nodes can still assemble a network and communicate each other while they have remaining mobile resources such as energy from batteries, Section 2.5.2. Routing protocols in MANETs are still facing the challenge of efficiently use of constrain resources from mobile devices.

In emergency scenarios, mobile resources are valuable. Victims, responders and outsiders expect the routing protocols utilised behind the communication of mobile devices consume less energy from batteries as possible.

Open source implementation To ensure reproducibility of data exchange and routing in a MANET, it is important to be able to reproduce the mobility traces followed by each mobile device and the exchange of messages between mobile nodes. Open source implementations are still limited, Section 2.5.4, and it is required to provide an open source code and well maintained environment to allow researchers experiment and test new approaches or features that can be reproduced and evaluated by peers.

4.2.2 Benefits of NDN to address limitations of MANETs

Although the NDN protocol is designed for wired networks, the advantages it may provide for MANETs are worth investigating [Meisel et al., 2010a]. This section presents relevant academic literature of NDN applied in MANETs, such as the case of Meisel et. al who observed that NDN potentially offers the following benefits, which make NDN particularly suitable for MANETs.

Routing Nodes in a MANET can use naming of data instead of IP addresses to communicate with each other. Nodes can directly forward the name of the content of interest to neighbours. Therefore, neighbours can reply with the respective content in case they have previously cached it. Consequently, the main benefit NDN provides is that nodes do not necessarily require IP addresses, and they do not need to precalculate single paths between nodes [Meisel et al., 2010a].

Caching NDN assigns a unique name to each content or to each segment of a content, which means that NDN packets can be distributed and cached by intermediate nodes. Content can be retrieved from any node that has cached it as caching is built into the network layer rather than being an application-layer concern. This is a difference from the traditional IP-based approach, where routers do not cache the information they have forwarded [Edens and Scott, 2017]. Caching not only reduces latency and reduces overall network traffic, it also ensures that content is available even if the origin node is not reachable. In NDN, every node on the network can serve as a cache, so caching can work in an infrastructure-free system such as a MANET [Meisel et al., 2010a].

Transportation As mentioned in Chapter 3, NDN is independent of the type of transportation. NDN enables communication on top of any network type as it forwards packets based on naming rather than host identities. In NDN, nodes are not required to be aware of the status of their connection to other nodes, which is a benefit in MANETs. It means that NDN suits both stable and intermittent network connectivity [Meisel et al., 2010a]. It also means that NDN networks can be layered on top of IP if desired.

However, while Meisel et al.'s work [Meisel et al., 2010a] makes a strong case for using NDN in MANETs, it does not address constraints such as energy consumption and has not resulted in a real open source implementation.

4.2.3 nMANET benefits to address MANET limitations

Although features of NDN clearly provide benefits for MANETs, name based communication is not fully exploited [Meisel et al., 2010a]. Today's mobile devices are equipped with internal resources with significantly more capacity than devices targeted by early research on MANETs. By utilising resources such as improved CPU power, memory capacity, flash storage and sensors, forwarding in MANETs can be enhanced. The following paragraphs describe how nMANET utilises these mobile resources in combination with NDN features to offer more effective and efficient data distribution in ad-hoc networks. This combination represents the key distinguishing feature of nMANET, going beyond the mere porting of NDN onto a MANET.

Forwarding strategies In MANETs, the location of mobile devices and their internal status constantly change, which makes a direct comparison with wired or static networks impossible. For example, remaining battery, available memory space and geographical location are not constant and are related to the conditions of the mobile devices at a particular point in time. nMANET takes into account the status of internal resources and the measurements from sensors to make forwarding decisions.

Compared with NDN, forwarding strategies in nMANET have a wider scope. In NDN, forwarding decisions are based on a single forwarding strategy and each forwarding strategy is based on the available entries in the FIB, which is either populated manually or through a separate process such as `nlsrd`, which implements the Named-data Link State Routing Protocol [Hoque et al., 2013] as a daemon. In contrast, nMANET divides the concept of forwarding strategy into two parts: *FIB next-hop selection* and *prefix discovery*. In both cases, the decision making process considers the status of local and nearby nodes.

Mobile resource awareness nMANET collects data about the state of mobile resources in neighbouring nodes. Forwarding strategies can utilise this data to make forwarding decisions that aim to best utilise available resources in order to improve efficiency, increase resilience and the lifetime of the individual nodes and the nMANET as a whole. Each node collects information only about its immediate neighbours and updates it reactively. These node status exchanges serve to discover neighbours, obtain status information as well as exchange information about the status of FIBs.

Energy awareness An important difference compared to NDN is that nMANET is aware of one of the most critical constraints in MANETs, remaining energy in batteries. nMANET intends to offer forwarding strategies aimed at minimising the energy consumption of nodes during the retrieval of the content. The aim to balance the interests of individual node owners to keep their mobile device alive while ensuring effective and efficient operation of the nMANET overall.

To achieve this, nMANET provides a methodology to estimate the amount of energy consumed by mobile devices that are using a particular forwarding strategy and compares the outcome to other existing strategies. This comparison provides a quantitative approach to rank all tested forwarding strategies for next hop selection and prefix discovery in nMANET. Details of these strategies and evaluation are detailed in Sections 4.5, 4.6 and 6.1.1.

Raw socket communication As NDN is not dependent on a specific type of network between nodes, NDN packets can be transported by traditional protocols, such as TCP and UDP using unicast or broadcast as well as protocols at lower levels such as the data link layer, using raw sockets.

Raw sockets allow direct exchange of packets between nodes without the need of any protocol-specific transport layer. Therefore, the aim is to carry NDN packets by using raw datagrams. Additionally, as NDN does not required the addresses of end points, the headers in raw datagrams can be omitted, this is because they are optional.

Technically, this can be implemented in Linux-based devices. However, in the case of Android operation system, unfortunately, it is not supported, which means that it does not provide access to raw datagrams. Consequently, in this thesis, this is considered future work as discussed in Section 7.1.

4.3 nMANET characteristics

This section describes the characteristics of nMANET, which were derived from an analysis of how the limitations of MANETs can be solved by applying NDN features. Figure 4.1 depicts the main modules of nMANET and the following paragraphs describe each of these nMANET characteristics.

4.3.1 Ad-hoc communication

nMANET links nodes in a mobile network in ad-hoc mode, which means that there is no central authority or infrastructure responsible for coordination. Nodes may have more than

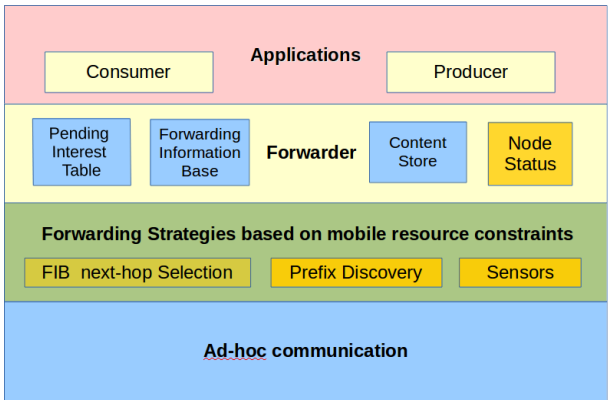


Figure 4.1: nMANET general modules

one network interface, and be connected to different networks, which could include network interfaces in infrastructure mode to connect to wider NDN-based or IP-based networks to ensure interoperability with existing infrastructures. For example, a mobile device can exchange NDN packets using one of its interfaces to communicate with a nearby node using Wi-Fi, at the same time using cellular radio to connect to an IP-based network. nMANET utilises mainly Wi-Fi interfaces due to their availability in most mobile devices and due to their wider coverage area and bandwidth compared with other technologies such as Bluetooth.

4.3.2 Applications

The role of consumer and producer is the same in nMANET as in a normal NDN network. This means that existing application software can be reused. Producers and consumers for nMANET running on Android would be written in Java using the JNDN library but in principle any NDN library for the chosen programming language can be used. A producer or consumer running on an nMANET node such as a mobile device would connect to a forwarder running on the same device, thus insulating the application software from the fact that it is running as part of a MANET with changing network connectivity. This is in contrast to the wired network case where a producer or consumer would connect to a forwarder that is part of the network infrastructure (like a router in an IP network).

4.3.3 Forwarder implementation in nMANET

The roles in an nMANET are similar to the three roles in any NDN network (see 3.4.1), with differences only for forwarders, which are adapted to operation in a MANET. nMANET forwarders utilise information about the status of the local node and neighbouring nodes to make forwarding decisions based on forwarding strategies. In other words, in contrast with

NDN, which was designed for static networks, nMANET needs to be aware of the status of the nodes in the network. This is required due to the mobility of nodes, the resulting changes in the topology of the network, and the changes of the status of internal resources of nodes, such as battery levels.

Compared with NDN, nMANET adds one additional table to cache status information for nearby nodes, which means that nMANET have four principal internal tables: the Forwarding Information Base (FIB), the Pending Interest Table (PIT), the Content Store (CS) and the Node Status (NS) table. While the first three are similar to those found in NDN forwarders, the node status table is exclusive to nMANET.

FIB The FIB contains prefixes with matching one-hop destinations that are likely to be able to satisfy interests with this prefix. Once an interest arrives at the forwarder and if it is to be forwarded, nMANET tries to match a prefix of the name with the available prefixes stored in the FIB. In the case that more than one entry of the FIB matches with the prefix of the incoming name, nMANET selects the next hop by applying a forwarding strategy to the list of matched entries. Therefore, nMANET is able to forward the interest packet to the next hop.

In the case that the prefix of the incoming name is not present in FIB, nMANET tries to discover whether nearby nodes have the prefix or not. This can be achieved by broadcasting a node status request, which is utilised by nMANET to retrieve up-to-date information from neighbours, including FIB prefixes. Node status packets are utilised to request information that describes the current status of a node. The information carried by the replies to node status request updates the FIB table.

In the case that the up-to-date FIB can satisfy the name of interest, then a FIB next-hop selection is executed to identify next hop or hops. Finally, in case the FIB entries from the neighbours do not suggest a feasible next hop, nMANET broadcasts the interest packet. Due to the mobility of nodes, by broadcasting the interest packet to neighbours, nMANET seeks to disseminate the interest to the neighbours of neighbours, spreading it throughout the network until a forwarder or producer is found that can provide the content. Section 4.5 presents more details about the strategies available for FIB next-hop selection and prefix discovery in nMANET.

PIT The Pending Interest Table is a table that maps interests to the nodes that have requested them. Its operation is similar to that in NDN. This table is utilised by nMANET in the reverse path content retrieval. When content arrives at the forwarder, it needs to know the list of nodes that have requested the incoming content. Therefore, an nMANET forwarder can satisfy pending requests by sending the content back to them.

CS Similarly to NDN, the Content Store in nMANET is a map of the name of the content to the content itself. It represents the cache of the node, which is used to satisfy future request with the same name. As memory (both RAM and flash memory) is limited in a mobile device, the content store will need to replace old content to accommodate incoming content. JNFD implements the LRU cache replacement mechanism. The implementation and evaluation of more complex, cooperative caching mechanisms is a research topic in its own right as it involves a trade-off between the benefits an individual node gains versus those for the whole network (cf. e.g, [Wang et al., 2017]).

Node status The node status table is a map of the identifier of a node (such as its IP address when using IP or its MAC address when not) to its most recent status reported through a node status packet. This table stores status information of the node such as location, battery status, or space available in the CS but also information from its FIB. As node status information is time dependent, it is aged out of the node status table after a period of time. New node status information is requested via a broadcast whenever an entry is found to have exceeded a configurable time limit used to define "freshness". An entry that has exceeded the freshness threshold but is not considered to be expired can still be used by the forwarding strategy but a request for new statuses will still be triggered. The status information is utilised by nMANET to make forwarding decisions as described below.

NACKs vs. Timeouts Negative Acknowledgements (NACKs), introduced in Section 3.4.3.5, can be used to signal to a requester that content with a given name could not be found. NACKs can be beneficial for notifying consumers about cases such as forwarding failures or requests for incorrect or unavailable information. However, it opens security issues, for example, it facilitates Denial of Service attacks [Compagno et al., 2015]. An alternative is to simply let interests time out if content for them cannot be found. This potentially reduces the level of network traffic. This is the approach taken in nMANET.

4.3.4 Resource-aware forwarding strategies

As discussed above, forwarding strategies in nMANET consist of FIB next-hop selection and prefix discovery.

FIB next-hop selection requires that a prefix of a name of interest is present in at least one entry in the FIB, which means that there is information of at least one next hop that could potentially have or be on a path to the content requested. The decision of which next hop to use if there is more than one option is made by the forwarding strategy on the basis of data about neighbouring nodes collected in the node status table.

For example, in the case of the storage-based strategy, which is explained in Section 4.5.5, the interest packet is redirected to the in-range hop that has the most free storage space. Another criterion that can be used by a forwarding strategy is the comparison of data received from neighbours with sensor data measured locally, such as latitude and longitude. In this example, an nMANET node can calculate the distance to neighbours and forward the interest packet to the geographically closest one or to one geographically closest to the node previously used as a next hop for a prefix that has since become unavailable.

On the other hand, in the case that a prefix of the name of interest is not present in the FIB, an nMANET node starts the prefix discovery process, which is to request updated node status information from its one-hop neighbours to see if any of them have the prefix in their FIB.

Once the prefix discovery is triggered, there are two possible outcomes: the prefix is discovered or it cannot be discovered. In the first case, the nMANET node that started the discovery process updates its local FIB and the FIB next-hop selection process is executed in order to select the next hop. In the case that the prefix cannot be discovered, the interest packet is disseminated to the one-hop nearby nodes, which start a similar cycle with their own neighbour nodes. The FIB next hop selection and the prefix discovery are discussed in more detail in Section 4.5 and Section 4.6.

nMANET forwarding strategies can be classified as reactive, as forwarding decisions are made at the time a request arrives at any node and prior knowledge of an up-to-date status of the network is not required. These characteristics reduce network traffic and mobile network resources are used only when a request needs to be satisfied. This is an important difference with deployed protocols of NDN such as NLSR presented in Section 3.4.7.1. Other differences compared to NDN are the additional features of nMANET presented in Section 4.7. These features were created due to the inherent mobility characteristics of nodes in a MANET.

4.4 Interest and data packet processing

As nMANET intents to be compatible with NDN, nMANET also utilises the NDN interest and data packets that were described in Section 3.4.2. However, as nMANET has introduced new features, packet types and tables, it also needs to modify the logic of exchange of packets between nodes. Figure 4.2 and Figure 4.4 depict how nMANET treats interest and data packets when they arrive at a forwarder. Note that these packet processing represents one important difference compared with the traditional NDN packet flow presented in Section 3.4.4.

In the case of an interest packet, nMANET differentiates between interest packets that contains a

request for a specific content and packets that contains control commands such as registration or node status requests. Section 4.4.1 presents these details and the decisions that nMANET makes in cases such as naming comparison against information of internal tables and how forwarding strategies are applied when the requested content is not available locally.

Similarly, and in the case of data packets. nMANET makes a distinction between data packets that carry specific requested content and packets that carry control commands such as node status. Section 4.4.2 presents in detail all these decisions and how forwarding information is updated into internal tables in order to be used for further request.

4.4.1 Interest packet processing

Figure 4.2 describes how interest packets are processed in nMANET. It starts when an interest packet arrives at a forwarder and nMANET verifies whether the interest is a control command. If it is a control command, then it verifies whether is it a prefix registration request or a node status request. In the first case, nMANET updates FIB entries, and replies to the requester with a registration acknowledgement. In the case that it is not a prefix registration request, then it corresponds to a node status request, which means that neighbours are requesting the forwarder to provide information about the status of the node. In this case, the forwarder collects information about the status of the node and sends it to the requester.

In the case that the name is not a control command, nMANET checks whether the name is already in the local cache (CS), in which case the content is retrieved from the local cache and sent back to the requester. In this case, the name is not inserted in the PIT as the name has already been satisfied by sending the cached content.

If the content of the name was not cached in the CS, nMANET checks if it was already requested by other nodes by looking whether the name is present in the PIT table. In the case that it is present, nMANET verifies that the interest packet does not correspond to a loop by checking the nonce. If the interest packet does not correspond to a loop, the PIT is updated.

Note that nMANET utilises the nonce value in combination with the name to identified loops. In the case that a second name and nonce arrives and has the same values as the ones stored in the PIT, this interest packet is considered as a loop and it is discarded.

Finally, having a name that was not cached in CS and that is also not present in the PIT causes nMANET to conclude that the name is new for the forwarder. The PIT gets updated and the interest packet is forwarded using the active forwarding strategy.

The following paragraphs highlight two particular cases in Figure 4.2. These two cases

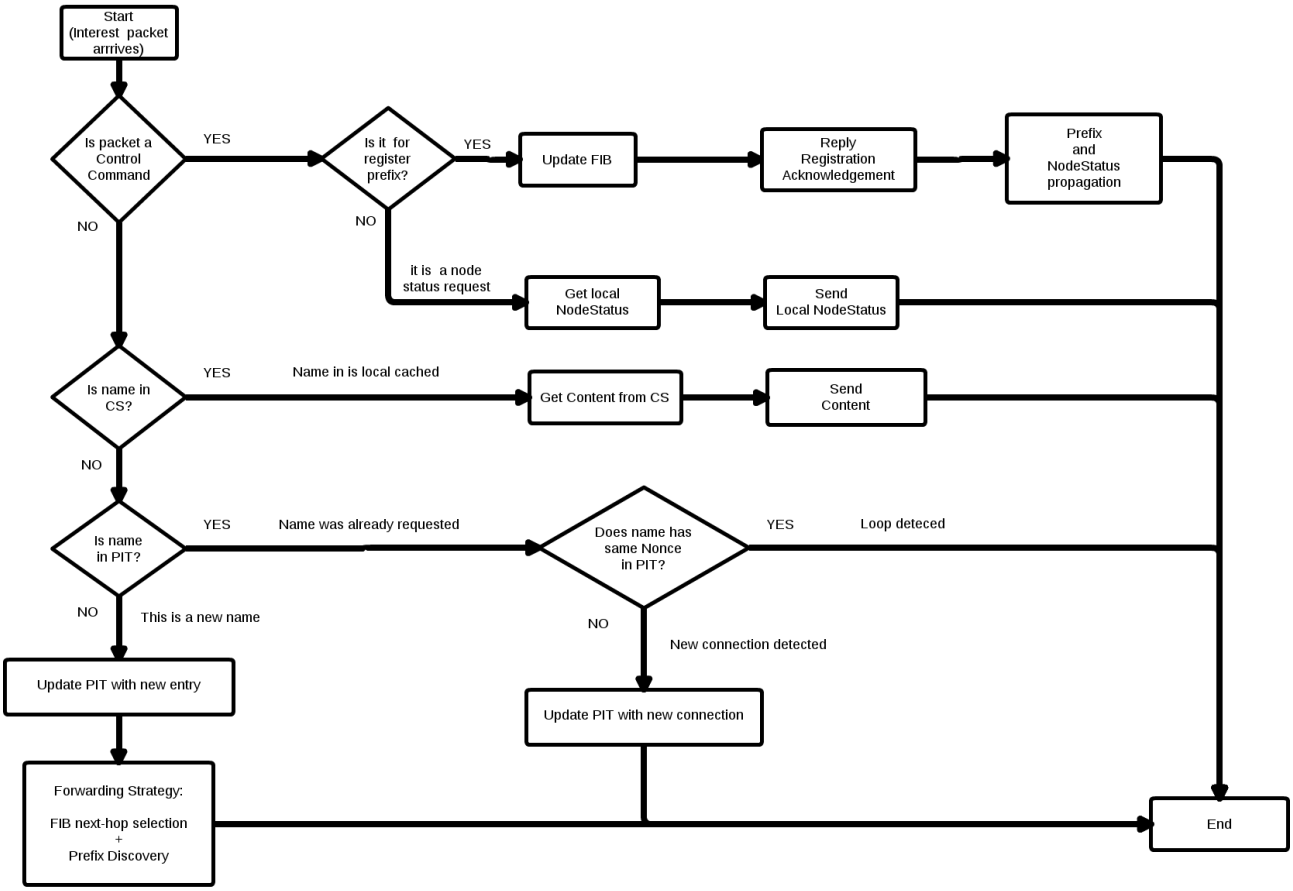


Figure 4.2: Interest packet flow in an nMANET forwarder

correspond to the outcomes after checking whether the name is a registration request or node status request.

Prefix registration in the interest packet flow A registration packet is a type of control message that contains the prefix of the content available in a producer. In the case that a registration packet arrives at the forwarder and nMANET finds that the prefix is already in the FIB table, only the connection from where this packet came is updated in the corresponding entry for this prefix. On the other hand, if the prefix is not present in the FIB table, nMANET adds a new entry to FIB as a pair of the prefix and the connection from where the registration packet comes. Like NDN, nMANET replies to each prefix registration request with a registration acknowledgement.

Node status in the interest packet flow Node status packets are a type of control message introduced by nMANET. Node status packets are interest and data packets with a dedicated

name, which is ”/localhop/nfd/rib/nodestatus”. In nMANET, a forwarder broadcasts node status request packets to nearby nodes to request information about the status of the node. This request is an interest packet under the name “/localhop/nfd/rib/nodestatus”. On the other side, each receiver of a node status request collects its own local node status information and loads it into a data packet with the same name. This node status response is then sent to the requesters.

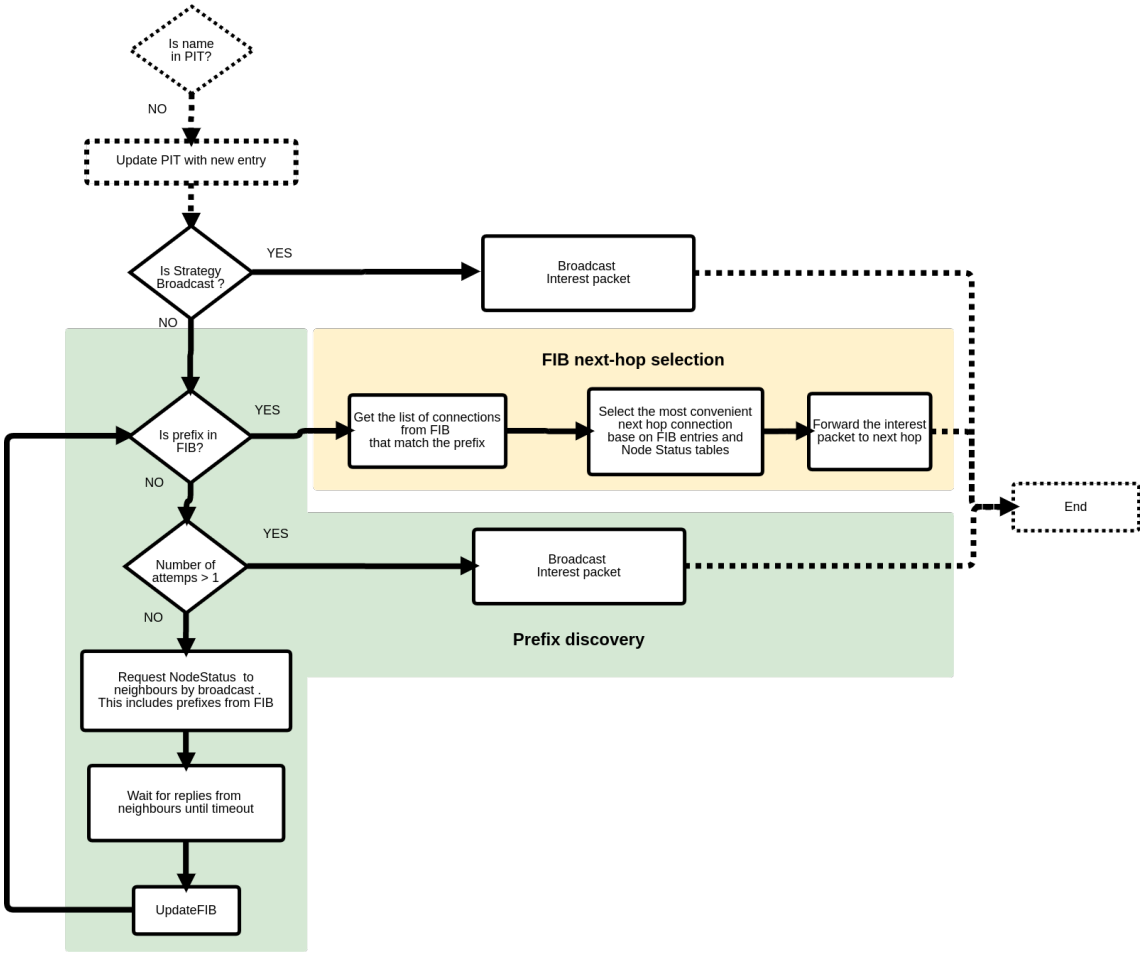


Figure 4.3: Flow of the nMANET forwarding strategy for interest packets.

Figure 4.3 shows how the next hop selection and prefix discovery work together to effectively and efficiently forward interests to nodes – fowarders or producers – that can satisfy the interest. Sections 4.5 and 4.6 describe how a forwarder utilises node status information from neighbours in order to select the next hop and discover available prefixes in nearby nodes.

4.4.2 Data packet processing

Figure 4.4 shows how a data packet is processed in an nMANET forwarder. nMANET checks if it is a control request response packet, which in this case means that it is a node status response

that is then used to update the node status table and the FIB. The FIB is updated with prefixes contained in the node status and with the sending node as the next hop. On the other hand, in the case that the name is not a control command, the name of the content is compared with the entries in the local cache (CS). If the name is already in the CS then the data packet is a duplicate and can be discarded.

If the name of the data packet is not present in the CS, nMANET verifies whether this name matches the entries in the PIT. From this comparison, there are two possible outcomes: one outcome is when the PIT does not contain the name, in which case the data packet is considered as not requested. However, it is cached as it can be used for further requests in cases such as the one discussed in Section 4.7.7. The second outcome is in the case that the name matches one or more of the entries in the PIT, in which case nMANET executes the following: updates the FIB and CS with the prefix and the content respectively, retrieves from PIT the connections towards the nodes who requested this name, and after sending the data packet to the nodes that requested the content, nMANET clears all matching PIT entries.

4.5 FIB next hop selection

Chapter 3 observes that NDN was not designed for wireless networks or for MANETs, it was designed for wired networks. It was also concluded that the available NDN forwarding strategies for MANETs, such as the proactive protocol of NLSR described in Section 3.4.7.1 and LFBL protocol described in Section 3.4.7.2, do not exploit all available internal resources in mobile devices. Mobile devices are equipped with more resources than traditional wired devices, for example, smart-phones, in general, include GPS receivers, accelerometers and battery sensors. nMANET utilises these resources to optimise forwarding decisions to retrieve information from producers. One key aim is to minimise utilisation of producers and dependency on them.

nMANET offers a different perspective on deploying forwarding strategies than NDN. nMANET forwarding strategies utilise inherently available resources in each node. For example, when a node makes a forwarding decision, the decision can be supported by information regarding the levels of remaining energy in batteries, the geographical location of mobile devices or other sensor measurements. While NDN makes forwarding decisions purely based on the entries available in the FIB for a particular prefix, in contrast, nMANET selects the next-hop by filtering the entries in the FIB and by considering the available resources of in-range mobile devices through the exchange of node status information.

The following paragraphs present the forwarding strategies available in nMANET to provide FIB next-hop selection. To explain them, a simplified and static configuration is used involving

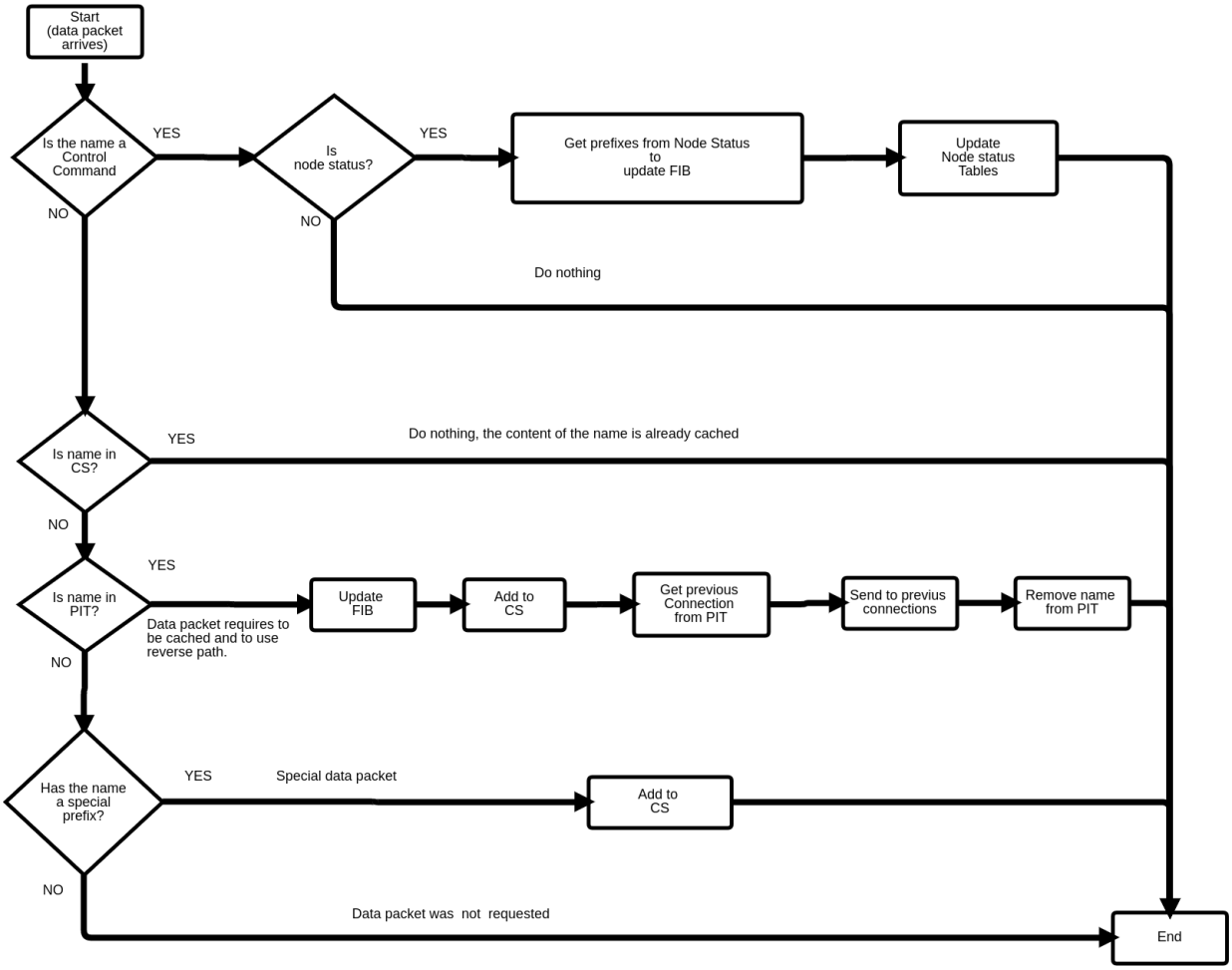


Figure 4.4: Data packet flow in an nMANET forwarder

a single consumer, a forwarder and two producers registering the same prefix. This is due to the objective of these paragraphs is to describe the forwarding strategy itself, and highlight the instant when a node makes a forwarding decision between potential next hops. To isolate the explanation, the initial assumption includes that the forwarder has already received a registration interest packet from the producer and that the prefix was successfully registered in the FIB of the forwarder. In each case, the consumer sends an interest packet matching the registered prefixes of the two producers, forcing the forwarder to choose between two possible next hops.

4.5.1 Unicast strategy

In case an nMANET forwarder utilises the unicast strategy, it first gets from FIB the list of connections that match a prefix of the name of interest. Then the forwarder resends the interest packet to all connections sequentially. For example, in Figure 4.5, as two connections match the

prefix "/a/b", one in P1 and another in P2, the forwarder sends the interest to P1, and then to to P2.

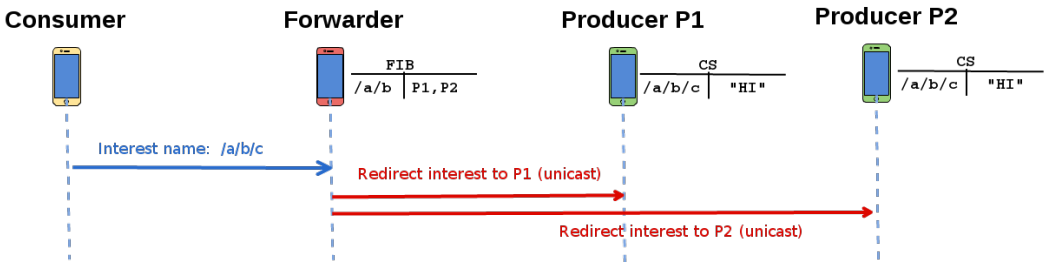


Figure 4.5: Unicast strategy

4.5.2 Broadcast strategy

The broadcast strategy in nMANET uses the broadcast feature of the Wi-Fi chip to send the interest packet from a forwarder to all other in-range forwarders, producers or nodes who are able to receive broadcast packets.

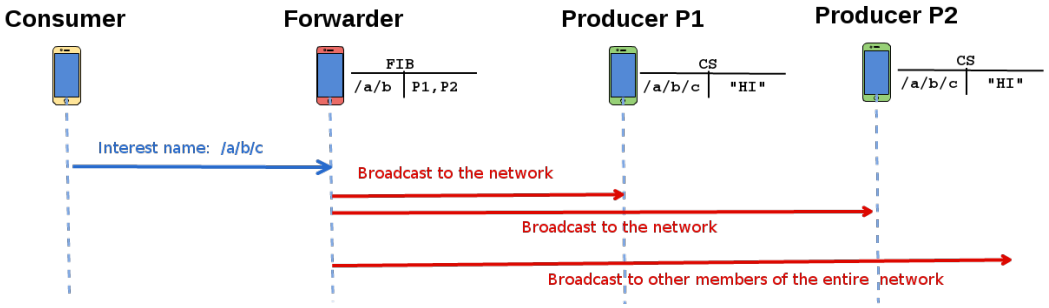


Figure 4.6: Broadcast strategy

nMANET utilises this strategy as one of the most basic methods to retrieve content from nearby nodes. As shown in Figure 4.3, in the case that a forwarder cannot satisfy the incoming interest packet and has no possible next hop in its FIB, nMANET immediately broadcasts the interest packet without requesting node status information from nearby nodes.

If the FIB table does contain prefixes matching the incoming interest, the broadcast strategy uses the unicast next hop selection to send the interest on to all suitable forwarders.

4.5.3 Random strategy

In the random strategy, the interest packet is forwarded to a next hop that is randomly chosen from a list of possible next hops that are stored in FIB. While this potentially reduces the amount

of unnecessary traffic compared to the unicast strategy, it also potentially fails to retrieve the content, leading to a timeout on the consumer side and causing it to re-issue the interest or give up.

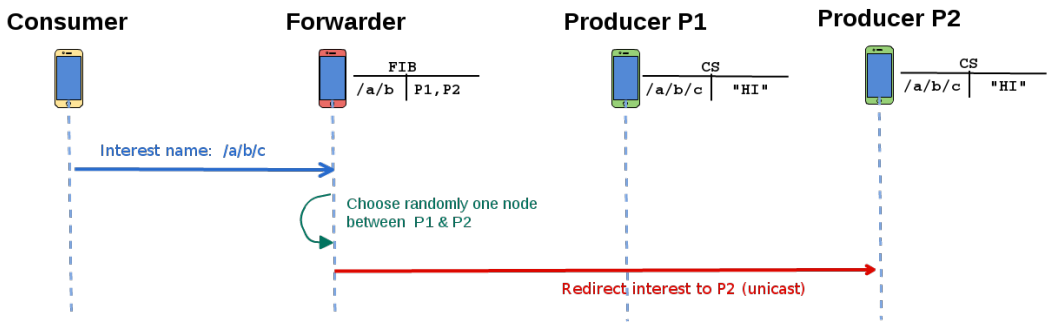


Figure 4.7: Random strategy

4.5.4 Geo-location strategy

By knowing the geographic location of neighbours in a network, forwarders can be aware of nodes in their proximity that potentially could become their next hop. However, as power consumption has a direct relationship with the radio range of coverage, to cover a large distance, the Wi-Fi transmission module requires utilising more power [Nugroho et al., 2014]. nMANET introduces the geo-location strategy as a strategy to minimise energy consumption.

The geo-location strategy minimises the transmission power by forwarding interest packets to the physically nearest forwarder or producer, assuming that neighbours agree to share their location when required. Geographical location collected from GPS sensors provide three coordinates: longitude, latitude and altitude. Based on these coordinates, two nodes can estimate the distance between them.

To estimate the closest node, the forwarder gets the list of nodes with prefix matches for the interest sent by the consumer. If node status information is not up to date, a node status request is sent. Using geo-location information, nMANETs estimates all respective connections based on their distance from the forwarder. nMANET ranks the nearby nodes by the distance from the forwarder, and forwards the interest to the neighbour with the shortest distance, as shown in Figure 4.8.

A limitation of the geo-location strategy is the high dependency on GPS receivers, which means it cannot be used in mobile devices without such equipment, such as laptops, or when GPS is turned off by the user. Another limitation is that the GPS sensor in mobile devices consumes more energy than other sensors. The impact can be see in the experiment reported in Section

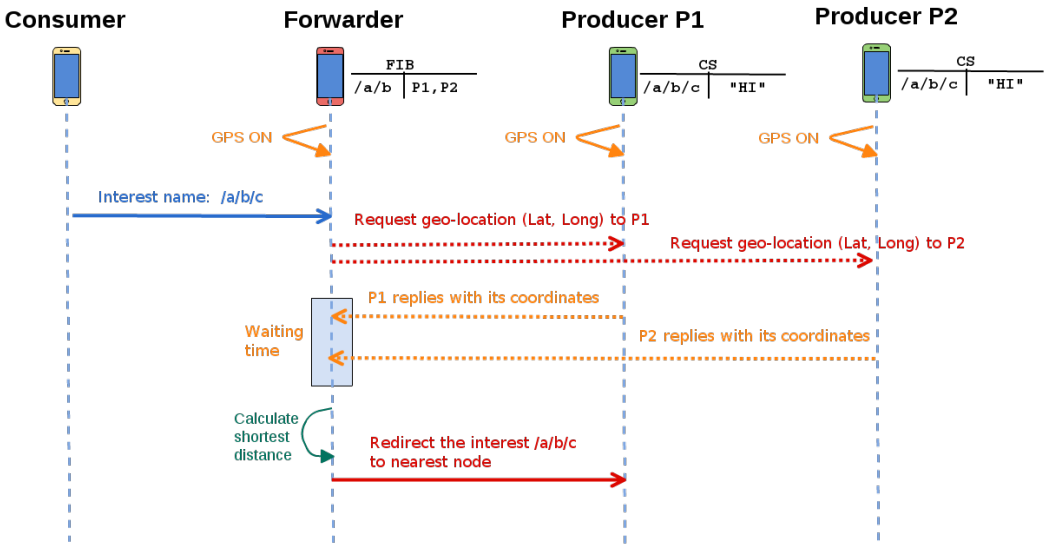


Figure 4.8: Geo-location strategy

6.5.3. If the GPS sensors are turned on anyway, as is likely in emergency scenarios, the use of this data for forwarding does not impose an extra cost, so should be taken into consideration.

To minimise the energy consumption of GPS receiver, an alternative approach is the use of the Receive Signal Strength Indicator (RSSI), that is a measurement of the level of signal strength from an access point or router. However, due to physical properties such as reflection and scattering, RSSI is a poor distance estimator **??**. For example, in an emergency scenario, the RSSI can vary depending of the physical structure and type of obstacles between mobile devices. Additionally, in the case that energy suppliers are unavailable, access points and routers become unavailable as well.

4.5.5 Storage strategy

Modern smartphones have storage space in the order of gigabytes in the form of built-in flash memory and some can even be extended with additional storage space via plug-in SD cards. This space can be used to back the content store of JNFD. However, as the size of the remaining storage constantly changes, being aware of the available size of the storage of a mobile node gives an idea of the ability of a node to cache furhter content. Consequently, a forwarder can compare the remaining sizes of storage of nearby nodes to decide which neighbour is the most suitable to forward an interest packet to.

In the storage strategy, the forwarder utilises information about the size of available storage of nearby nodes, which is used to decide which next-hop the interest should be forwarded to. The basic premise of the storage strategy is that a node with sufficient free storage is a potential

candidate to cache more content and handle more requests.

In this strategy, first the forwarder gets a list of nearby nodes that match a prefix of the name of interest. Second, based on this list, the forwarder gets information about the size of the available storage of these nodes, sending node status requests if needed. Finally, the forwarder redirects the interest packet sent by the consumer to the node with the most available storage.

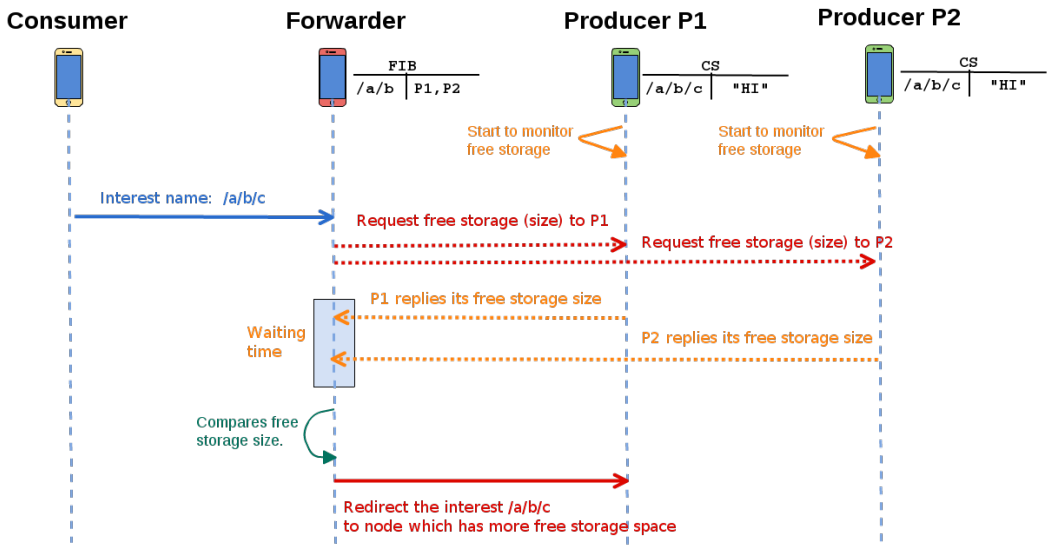


Figure 4.9: Storage strategy

4.5.6 Battery strategy

As discussed in Section 2.5.2, to maximise the network lifetime, it is necessary to minimise the utilisation of remaining power in batteries. Therefore, monitoring of the remaining energy in the battery is required to make forwarding decisions.

Similarly to the storage strategy, in the battery strategy, the interest is forwarded to the node with the most remaining battery capacity as this protects the node with less power from running out of energy, rendering it useless for the user but also removing it and its resources from the network. nMANET uses the battery monitoring functionality built into the Android operating system to gather information about battery life[1].

Figure 4.10 shows that by using the battery strategy to select the next hop, the forwarder first gets a list of possible next-hops with matching prefixes. Based on this, the battery status of the candidates is retrieved, sending node status requests where necessary. Finally, the interest packet sent by the consumer is forwarded to the next hop with the most remaining battery.

[1]https://developer.android.com/training/monitoring-device-state/battery-monitoring.html

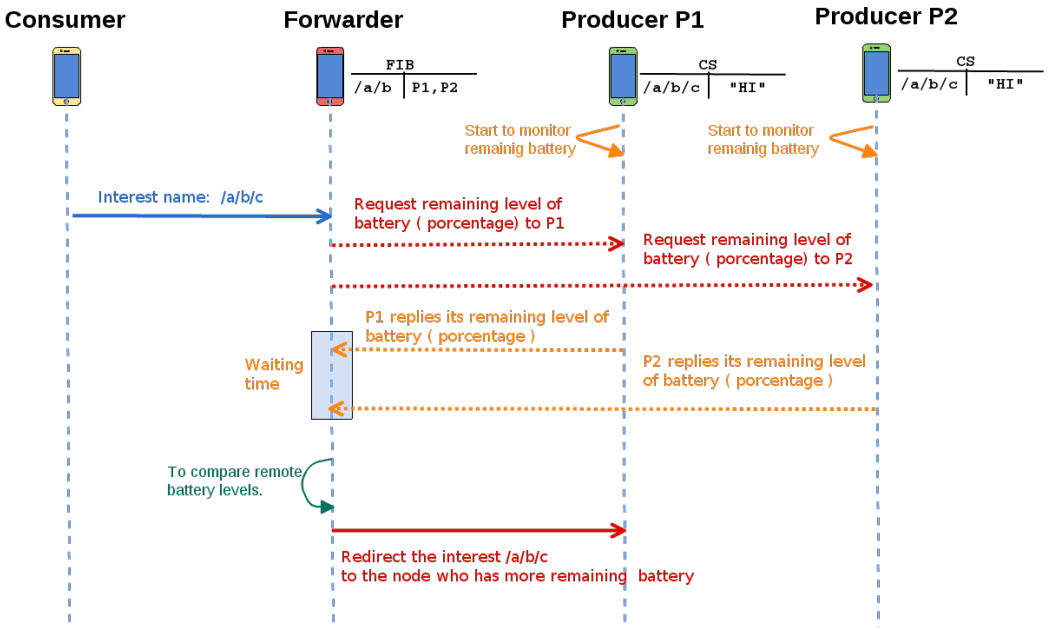


Figure 4.10: Battery strategy

4.6 Prefix discovery in nMANET

In the case that the required prefix is not present in the local FIB, it is necessary to discover the prefix in a way that the forwarder can redirect the interest packet. Based on the details presented in Section 3.4.7, Table 4.2 summarises the available NDN approaches that aim to address this problem. The problems that these approaches address include discovering prefixes, configuring the FIB, and forwarding an interest that does not match an entry in the FIB.

As most of the approaches show in Table 4.2 were designed to be applied in wired networks, nMANET uses its own prefix discovery approach as illustrated in Figure 4.3. In the case that the prefix is not present in the local FIB, nMANET tries to find out whether this prefix is present in the FIB of one-hop nearby nodes using the node status mechanism. The forwarder can update its own local FIB with the replies from nearby nodes and check again whether the required prefix is now present or not. In the case that the required prefix is present, nMANET applies the FIB next-hop selection as described in Section 4.5.

On the other hand, in the case that the prefix is not present in the FIBs of one-hop nearby nodes, nMANET broadcasts the interest packet to all in-range nodes. The main reason to disseminate the interest packets is so that nearby nodes can then repeat the same procedure with their own local FIB and with their own neighbour nodes. Therefore, the interest packet is disseminated towards more nodes that may have the content or have a next hop in their FIB so that the interest can be sent on using an established path.

Approach	Description
Broadcast-based self-learning	The consumer broadcasts only the first interest to update entries in intermediate FIBs by observing the reverse path of the data packet. Then, further interests with the same prefix just need to use unicast.
Simplified Content Delivery Network	Flood messages to create DAGs, which contain the next hop of a node.
Reactive Optimistic Name based routing	Similar to Broadcast based self-learning.
Listen First Broadcast Later	Broadcast the interest only when no other nodes are sending the same interest.
NLSR	Frequent exchange of routing information between nodes populates the FIB. It is proactive protocol.
Neighbourhood-aware Interest Forwarding	Forwards interest packets based on forwarding statistics. It does not explain how to populate empty FIBs.
Best Route Strategy	Send the interest to the lowest-cost next hop, and if it is not satisfied, it forwards to the next unused hop.
Multicast strategy	Flood the interest to a group of next hops available in the FIB. It does not cover FIB configuration.
Client Control Strategy	The consumer decides to which next hop the interest will be sent. It requires configured FIB entries as it does not discover prefixes.
NCC	FIB entries are updated manually like static routing tables in IP.
Access Router Strategy	It is similar to broadcast-based self-learning.
Auto prefix propagation	In cases where a single connected gateway forwarder exists, a forwarder can push registration information towards it, allowing prefix registration information to be disseminated.

Table 4.2: NDN's methods to discover prefixes, configure FIB or retrieve the desired content (cf. Section 3.4.7).

Reverse path FIB updating Following the broadcast of an interest, a forwarder will be waiting for data to arrive back via a path discovered, with the Interest sitting in the PIT. Once data flows back from one of the forwarder's neighbours, it can enter that neighbour into the FIB as a next hop for the name.

At booting time, all forwarders have empty FIBs except for the one that the producer registers the prefix with. In this example, when a consumer requests content for name /a/b, the intermediate forwarders execute prefix discovery by broadcasting the interest packet to neighbours until it eventually reaches the forwarder that has registered the prefix from the producer. At this stage, even though the content was found and can be delivered to the consumer, the reverse path is not recorded in the FIBs of forwarders as the reverse path data delivery only requires information from PITs. Consequently, further interest packets with the same prefix would need to be broadcast again. nMANET solves this limitation by updating FIB tables of forwarders when a content is retrieved using the reverse path.

To update the FIB table of each intermediate forwarder in the reverse path, nMANET extracts the name from the data packet and verifies whether the prefix of the name is in the FIB. In the case that it is in the FIB, nMANET verifies whether the associated connection is already present in the table, which subdivides the scenario into two sub-cases. One sub-case is when the connection is already present and nMANET does nothing as next hop for this prefix is already present in FIB. The second sub-case is when the connection is not the present in FIB, nMANET adds the new connection to the same prefix in the FIB. Finally, in the case that the prefix is not present in the FIB, nMANET adds the prefix and the connection as a new entry, that can be used to redirect future requests and eliminates the need of broadcasting for this prefix.

Note that the difference of this approach with the Broadcast self-learning presented in Section 3.4.7 is that Broadcast self-learning requires the consumer to broadcast the first interest in order to update the intermediate FIBs. In nMANET this mechanism is autonomous, it is part of the forwarding strategy, and does not required be triggered by the consumer.

A new node joining an nMANET will not have any FIB entries and would need to rely on broadcasts to forward interests. Section 4.7.3 describes a mechanism for populating the FIB by making node status requests. This nMANET feature contributes to minimising the use of local resources and network traffic as well as to speed up the transfer of interests through the new node.

4.7 Additional features of nMANET forwarding strategies

Besides the core functionalities, presented in previous sections, on how interest and data packets are treated, the following paragraphs describe additional features of forwarding strategies in nMANET. Table 4.3 presents a brief summary of when to use each feature and whether it is implemented in the nMANET prototype presented in Chapter 5.

Feature	Implemented?	Description
Limiting broadcasts	Yes	Controls the dissemination of non-satisfied interest packets. This can be used to reduce the overall use of resources of the network.
Granularity of prefixes	Yes	This is used to reduce the number of entries in the FIB. It reduces the size of the FIB. Consequently, a forwarder uses less remaining mobile resources such as memory and battery.
Initial FIB configuration	Yes	This feature is generally used at booting time, as it allows new joiners to become aware of FIB information from nearby nodes.
Prefix propagation	Yes	This is used to disseminate information from the producers towards the consumers. For example, to disseminate the list of available shelters in an emergency area.
Gateway	No	This feature is used to search information outside the MANET. When the required information is not available inside the MANET, a request is sent to a gateway.
Leaving notification	No	In the case that a node is close to leaving the network, this feature shares relevant information to nearby nodes, such as content and routing information.
Data push	Yes	Pushing data to forwarders, allows them to store standard information from the producers. For example, producers can push the list of available shelters, so that forwarders can deliver them to consumers in need.
Maximum waiting time	No	This is useful when the forwarder sets the level of urgency of a request.

Table 4.3: Summary of additional features of nMANET

4.7.1 Limiting broadcasts

The worst case for the prefix discovery strategy described above is that the whole network is flooded with the interest. This will happen at the beginning, when an nMANET is started and FIBs are empty but over time should happen less and less often as FIBs contain usable next

hop information. Forwarding strategies in nMANET control the dissemination of non-satisfied interest packets by limiting the number of broadcasts. The information about how many hops an interest has made via broadcasts or the number of broadcasts left has to somehow be carried with the interest.

The following paragraphs discuss possible solutions and their limitations. One initial solution is to carry the information that limits the number of broadcasts as an additional component appended at the end of the name of the interest, for example "/a/b/c/4times". This appended value is incremented with each broadcast. However, this solution does not guarantee interoperability with other platforms such as NDN for wired networks.

A second alternative utilises the lifetime field of interest packets, reducing it at each hope until eventually it reaches zero, in which case the forwarders will stop broadcasting the interest packet. The problem with this approach is that it tampers with the intended use of the interest lifetime field.

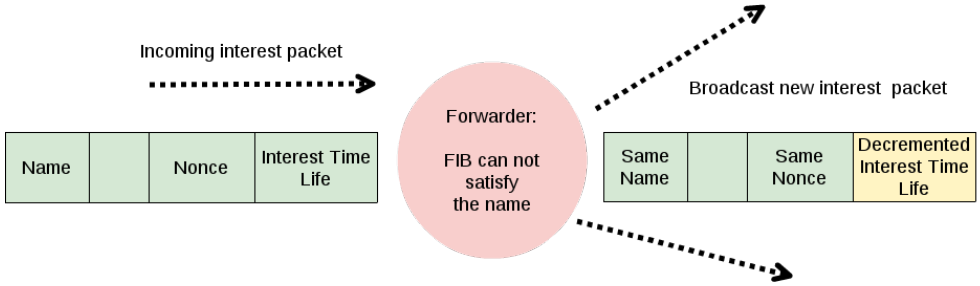


Figure 4.11: nMANET approach to limit and control the number of consecutive interest packet broadcasts.

A final alternative, implemented in nMANET, is to add the number of remaining broadcasts to the value of the lifetime of the interest packet. For example, in the case that an interest packet should be broadcast five times, the initial lifetime value of the interest packet is set to 4005 milliseconds, where 4000 is the original lifetime of the interest packet and five is the number of times to broadcast this interest packet. Therefore, the next forwarder decrements the lifetime value by one and broadcasts again. The lifetime is decremented every time a forwarder broadcasts the interest packet until it eventually reaches 4001, in which case broadcasting of the interest packet is stopped. Note that the value of 4000 is reserved for new interest packets such as the ones sent by a consumer at the beginning or by some other node outside the nMANET network.

A drawback of this approach as described so far would be that content more than five hops away would not be reached. This can be overcome by initiating the prefix discovery again with a higher value for the number of broadcast hops in a second attempt. This way, the number of broadcasts will still be limited in most cases. Another, practical problem when using the Java Client Library

of NDN (JNDN) is that by modifying the "interestLifeTime" field of an interest packet, a new nonce is generated. This is overcome in the implementation by patching the original nonce back into the interest packet. Figure 4.11 presents how nMANET solves this problem.

4.7.2 Granularity of prefixes

One problem that prefix discovery gives rise to is the potentially large number of FIB entries, granularity problem. As these are generated by interests, they consist of names for specific content requested. The number of these entries can grow to a significant size, which is particularly problematic on resource-constrained devices. It is therefore desirable to increase the *granularity* of FIB entries if possible. One possible approach would be to introduce FIB entries that attempt to collapse multiple fine-grained entries into a more coarse-grained one by identifying common prefixes.

The solution implemented in nMANET at the moment removes the last name element from the name before adding an FIB entry. This heuristic runs both the risk of over- and under-generalising but is a workable solution for the kinds of applications envisaged for nMANET and it has the advantage that it comes free of any cost in terms of network traffic. Another approach would be for a forwarder that a producer has registered with to disseminate information about what prefix was registered along the reverse path of an interest. This would require another control command, which is not currently implemented in nMANET. A proactive alternative to this reactive approach is described in Section 4.7.4.

4.7.3 Initial FIB configuration for new forwarders at booting time

When a new forwarder joins an nMANET, by default its internal tables such as CS, FIB and PIT are empty. In the case that a new interest packet arrives, the forwarder is unable to redirect it and needs to start the prefix discovery process by broadcasting node status requests. In nMANET, a new forwarder can obtain available prefixes from neighbours by two main procedures: reactive and proactive.

Forwarders that join for the first time can utilise the reactive procedure by waiting for incoming NDN packets disseminated by other nearby nodes. As there are two types of NDN packets, a forwarder can follow two different types of reactive procedures. One is the case when an interest packet arrives at the new forwarder. As its FIB is empty, it executes the prefix discovery described in Section 4.6. A second case is when node status packets arrive from nearby nodes. In this case, the new forwarder updates its local FIB with the prefixes carried by the node status packets, therefore the new forwarder becomes aware of prefixes that can be satisfied by neighbours.

On the other hand, and at nMANET booting time, new forwarders utilising the proactive procedure need to make only one request to all in-range nodes by sending a node status request. The node status returned by each neighbour contains the list of available prefixes from its FIB table. Therefore, new forwarders can update their own FIB, be aware of prefixes of nearby FIBs, and can apply any FIB next-hop selection strategy to redirect the interest packets to the next hop that potentially can satisfy the interest packet.

4.7.4 Prefix propagation

Forwarders in nMANET cannot only disseminate the name of interest from consumers, they can also disseminate the available prefixes from producers. The motivation for prefix propagation is to support dissemination of relief information during emergency scenarios, such as the list of places where Red Cross responders are located. Note that in this example, the responders are the producers, and they are interested in disseminating relief information to victims instead of waiting for requests. In emergency scenarios, there are more chances to have more victims than responders, which can be translated as more consumers than producers. Instead of letting the responders wait for incoming requests from victims, nMANET propagates the prefixes from producers as soon as they arrive at a forwarder.

Another important reason for prefix propagation is that victims are not aware of available relief information from responders. Consequently, victims are forced to guess possible available names, or find them through links contained in content they access, then try to see whether nodes in the network can satisfy them. By implementing a prefix propagation process, producers can not only ensure that content can be updated quickly, through established routes, but can also implement *server push*.

From a forwarder perspective, prefix propagation is reactive, in the sense that it is triggered by the prefix registration. nMANET propagates prefixes by broadcasting them to in-range nodes, and uses broadcasting due to the unpredictable mobility of nodes that causes constant changes in the topology of the network. Prefix propagation is an optional feature in the current implementation of nMANET, controlled by a command-line switch.

4.7.5 Gateway

The idea of a gateway in nMANET serves the same purpose as a gateway in IP-based networking. It connects independently managed networks up to form a larger whole, an internet, and allows the routing of traffic across the boundary. Gateways also often translate between different underlying networking technologies such as between Wi-Fi and Ethernet, or Ethernet and wide-

area networking technologies. To this end, they need to have network interfaces for the network technologies to bridge. In the nMANET case, this could be a 3G or LTE mobile communications module. As nMANET nodes will usually be smartphones, it is likely that in practice, a node will be a participant's smartphone or a device brought into the field by an aid organisation. In general, it could also be a device that connects to a wired infrastructure and that participates in the nMANET using a Wi-Fi network interface.

To act as a gateway, the device needs to route traffic between the nMANET and the outside network infrastructure. This involves identifying when content cannot be provided by the nMANET nodes and when interests therefore have to be forwarded to the outside network. Similarly, for content generated inside the nMANET (information generated about an emergency), this may need to be forwarded to the outside network. In most cases, the network connection to the outside infrastructure will be a limited resource and the gateway may need to ensure that traffic is not forwarded to it unnecessarily. In other words, the challenge of using gateways resides in forwarding the interest packet outside the nMANET only if the corresponding content is not available inside the nMANET. This challenge also involves finding an approach for nMANET nodes to stop gateways forwarding an interest packet when the required content is already in the cache of a forwarder or local producer inside the nMANET.

4.7.6 Leaving notification

This feature is employed when a node is about to leave the network, perhaps because its battery is nearing depletion or the user decides to take the device offline. In this case, the node can notify nearby nodes about its imminent departure and offer to hand over some of its state to them. Of particular interest are the FIB and the PIT, which could be used by neighbouring nodes to replace the leaving node with other possible next hops and to take over as intermediate nodes in paths for data matching a pending interest. The leave notification can also be used by the leaving node's neighbours to proactively seek new paths for any pending interests that have been forwarded to the node that is leaving.

4.7.7 Data push

In an emergency situation, it may be desirable to push content into the content stores of forwarders to reduce the time it will take for this data to be retrieved by consumers and also to protect against the producer becoming unavailable. While some data may be so important that it should be pushed to all nodes in an nMANET, some data may only be pushed to forwarders no further than n hops away. This feature is motivated by emergency scenarios where humanitarian responders, such as the Red Cross, aim to broadcast important and standard relief information to victims,

such as the location of hospitals or shelters. In this way, responders can disseminate essential humanitarian information to victims or they can improve access to this information. This feature in nMANET requires forwarders to cache data packets that were not requested. For security reasons, this feature may need to be limited to a small set of producers that are allowed to do this and that register special prefixes that are recognisable, such as "/relief/redcross".

4.7.8 Maximum waiting time for node status replies

As show in Figure 4.3, when nMANET starts the prefix discovery process, a forwarder requests status information from nearby nodes, including the list of their FIB prefixes. As the requester has no control over nearby nodes, neighbours can take different times to reply or decide even not to reply at all. nMANET sets a maximum waiting time for replies from neighbours, called a *threshold*, this is so that the node status request and response process does not excessively delay forwarding of the interest. A more sophisticated scheme would not work with a static threshold but would instead gather statistics about response times.

4.7.9 FIB updating logic

The FIB of a forwarder can be updated in different scenarios which include the following: receiving a registration packet (Section 4.4.1), obtaining the results of a prefix discovery (Section 4.6), obtaining the results of a node status request from neighbours (Section 4.4.1), data arriving back from a next-hop neighbour through reverse path data retrieval (Section 4.6) and performing the initial FIB configuration at booting time (Section 4.7.3). Figure 4.12 presents the logic that nMANET utilises to update entries in the FIB.

Initially, there are two restrictions on the FIB. The first one is that an FIB entry cannot have a connection from its local host. nMANET does do nothing when this case appears as it potentially comes from a broadcast generated by the same local host. Consequently, this loop connection should not be in the FIB. The second restriction is that the FIB and PIT cannot have simultaneously the same connection for the same prefix and name. This is to eliminating infinite forwarding loops between forwarders.

In the case that the packet does not come from a local host and it is not in the PIT, nMANET verifies whether the prefix is in the FIB. If it is not in the FIB, then it corresponds to a new forwarding information, for which nMANET creates an entry with the prefix and the connection and then inserts it into the FIB.

On the other hand, in the case that the prefix is already in the FIB, nMANET verifies whether the connection needs to be appended to the existing connections in the FIB for that prefix. To achieve

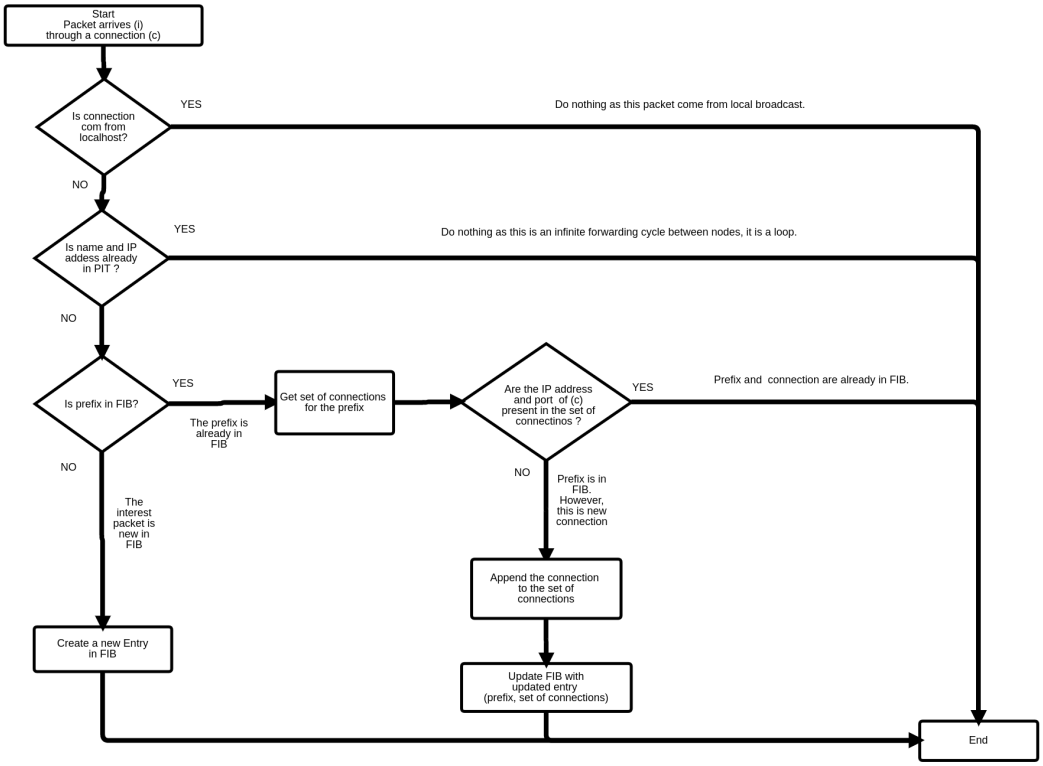


Figure 4.12: FIB update logic when an interest packet arrives from a connection (c).

this, nMANET extracts the set of connections of the prefix and verifies whether the address of the incoming connection is present in the set of connections. If the incoming connection is not present, nMANET considers this incoming connection as a new connection. This new connection is appended to the set of connections and then the entry is updated in the FIB. Finally, if the incoming connection is also present in the set of connections, nMANET only updates the FIB with this incoming connection if it corresponds to a connection with a different port. In other cases, nMANET does nothing as it considers the prefix and incoming connection are already in the FIB.

4.7.10 PIT updating logic

Figure 4.13 depicts how nMANET manages the PIT entries. When a new interest packet arrives, nMANET creates a new entry in the PIT only when the name of the interest was not already requested by another interest packet. In the case that the name is already in one of the entries in the PIT, nMANET checks whether the connection comes from an already known source based on IP address and port number. If not, then nMANET appends the new connection to the set of existing connections for the name of the interest.

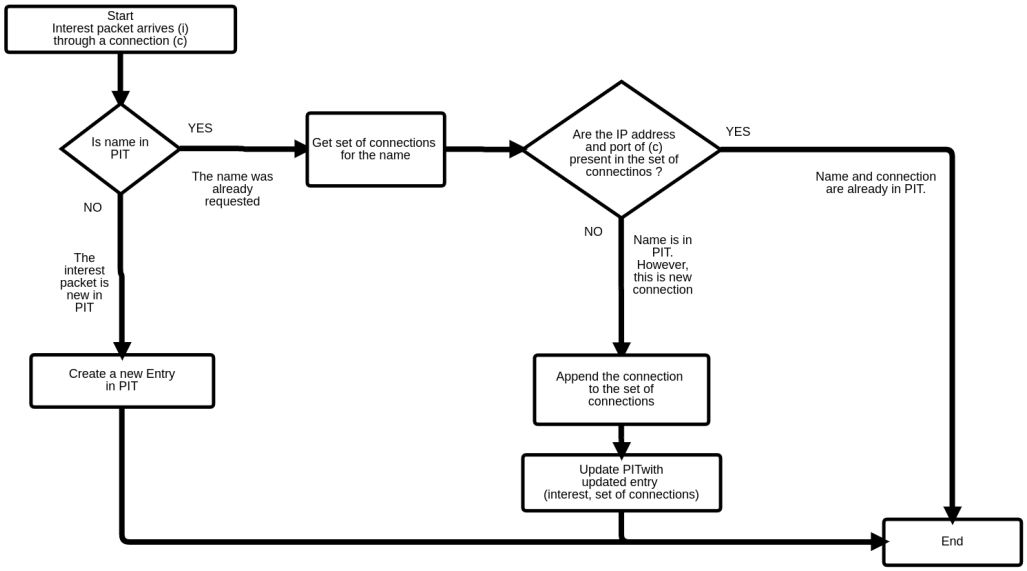


Figure 4.13: PIT update logic for when an interest packet arrives from a connection "c".

4.8 Chapter summary

This chapter has presented how features of NDN are utilised and adapted in nMANET to overcome the limitations in MANETs. The benefits and characteristics of nMANET are discussed, which include the flow of how nMANET processes interests and data packets, and how node status information is utilised and exchanged between nodes to make forwarding decisions in the forwarding strategies.

In addition, this chapter describes the scope of nMANET forwarding strategies, which include two main modules: FIB next hop selection and prefix discovery. It also lists and describes the additional nMANET features created to address the limitations of IP-based MANETs.

This chapter lists the minimum requirements for practical implementations. Chapter 5 makes reference to most of these as it describes the implementation of the Java NDN Forwarding Daemon (JNFD), a prototype of nMANET utilised for testing and experimentation in Android- and Linux-based devices.

CHAPTER FIVE

JNFD PROTOTYPE DESIGN AND IMPLEMENTATION

This chapter presents the Java NDN Forwarder Daemon (JNFD), which is a prototype implementation of nMANET for Android phones as well as Linux-based computers.

First, the functional and non-functional requirements are described, followed by a discussion of requirements arising from the need to evaluate JNFD for the purposes of this thesis. Following this, a number of key design decisions and the overall architecture of JNFD are discussed, followed by an outline of the object-oriented design of the classes that make up the system and the reuse of third-party libraries. The implementation process is described, especially the issue of testing and experimenting with JNFD. The chapter concludes with some notes on limitations of the current implementation.

5.1 JNFD Software design

This section presents the object oriented-design of classes and their relationships in the implementation in JNFD. This section follows the software engineering principles of object-oriented software design and implementation [Sommerville, 2010, p. 176].

5.1.1 nMANET main requirements

The functional and non-functional requirements to implement the nMANET approach are depicted in Tables 5.1 and 5.2, respectively. Both are expanded in the following paragraphs.

5.1.1.1 Functional and non-functional requirements

Functional requirements to implement JNFD are based on the discussions about nMANET characteristics of Section 4.3, nMANET packet flow diagrams of Section 4.4, FIB next-hop selection and prefix discovery of Section 4.2.3 and the additional features of the nMANET approach presented in Section 4.7.

The following list is a summary of how JNFD should behave in the case that an interest or data packet arrives at a JNFD forwarder, and table 5.2 contains a more technical oriented list of functional requirements of JNFD to interact with other nodes and its environment. The non-functional requirements are listed in Table 5.3, which presents the constraints of the functional behaviour of JNFD. Finally, reproducibility and scalability in nMANET are presented in the last part of this subsection.

1. **Interoperability**: JNFD should be able to receive packets from and send packets to NDN nodes. This requires JNFD to utilise the packets for interest, data, prefix registration and registration acknowledgement described in Section 3.4.2. For example, nMANET can exchange NDN packets with other NDN implementations such as NFD[1], Mini-NDN mulator [2] and ndnSim[3] simulator.

2. **Interest packet flow**: in the case that an interest packet arrives at a forwarder node, JNFD should treat the interest packet according to the diagram flow presented in Section 4.4.1.

3. **Data packet flow**: similarly, JNFD should treat incoming data packets according to the diagram flow presented in Section 4.4.2.

4. **Forwarding strategies**: in the case that an interest packet cannot be satisfied by a forwarder, JNFD should follow the two parts of the nMANET forwarding strategy presented in Section 4.2.3. Initially, JNFD should apply the FIB next-hop selection in the case that the name of the interest can be redirected by using the list of prefixes in its local FIB, as discussed in Section 4.5. In the case that the interest cannot be redirected by using the available prefixes in FIB, JNFD should start the prefix discovery process presented in Section 4.6.

5. **Additional nMANET features**: JNFD should implement additional features of nMANET as described in Section 4.7, which are required to make JNFD functional in MANETs. Table 5.1 presents the minimal features that should be implemented in order to have a functional prototype of nMANET.

[1] http://named-data.net/doc/NFD/current/INSTALL.html
[2] https://github.com/named-data/mini-ndn
[3] http://ndnsim.net/2.3/

Feature	Description
Limiting broadcast	Control broadcasting in the network when interest packets cannot be satisfied.
Reverse path FIB updates	Update the FIB table when data packets flow from producer to consumer.
Granularity of prefixes	Minimise the number of prefixes to be propagated in nearby nodes.
Initial FIB configuration for new forwarders	Configures clean FIB tables with nearby available entries.
Prefix propagation	Disseminates prefixes from the producer towards consumers.
Unexpected data	Content that was not requested. However it is important to cache locally, for example: emergency data from responders.
Matching and update logic for FIB and PIT tables.	Implement the logic of how to match and update entries in FIB and PIT tables.

Table 5.1: Minimal features to be implemented in a nMANET prototype.

Requirement	Description
Ad-hoc communication	1. Be able to establish mobile communications between nearby devices using Wi-Fi interfaces in ad-hoc mode.
Tables	2. Each forwarder should have the following tables: PIT, FIB, CS and NS (Figure 4.1).
Manual settings	3. JNFD should offer manual operation for initial settings purposes, such as the case of enable/disable prefix propagation.
Strategy	4. Offer forwarding strategies based on available internal resources and sensors, such as GPS, battery, storage, etc. This includes FIB next-hop selection and prefix discovery. 5. Manual initial selection between strategies for experimentation purposes.
Transport	6. Provide transportation through UDP and TCP, and if possible raw Ethernet packets. 7. Manual selection of the type of transportation.
Node Status	8. Be able to get and exchange local node statuses, which include the list of FIB prefixes.
Consumer	9. Let consumers send an interest packet(s) to a specific node or group of nearby nodes through unicast or broadcast.
Producer	10. Let producers register prefixes in a specific node or a group of nearby nodes through unicast or broadcast.

Table 5.2: Main nMANET technical oriented functional requirements to implement JNFD

Requirement	Description
General	- Run in background as a service as it should be transparent to the user and the application. - Run in most of Linux based mobile devices such as Android smart-phones, Linux-based laptops. - Run in simulators such as Mininet-WiFi in ad-hoc mode [Fontes et al., 2015].
Logging	- Send logs to standard outputs (console, screen), and store them in internal files.
Compatibility	To be compatible with actual NDN infrastructure for wired networks.
Performance	To be capable to measure performance when data is retrieved. For example using metrics such as completeness and correctness.
Cost	To be capable to measure involved costs when data is retrieved. For example retrieve data using metrics such as energy consumption, cpu usage, and memory usage.

Table 5.3: Main nMANET non-functional requirements to implement JNFD

5.1.1.2 Reproducibility

Reproducibility is understood as the ability to reproduce results based on the same input data and computer program. It is an important cornerstone in computational experiments that allows computational results to be independenty verified. A study of 601 papers published in top ACM conferences and journals conducted by Coolberg et. al (2015), illustrates the importance and difficulty of reproducibility in Computer System Research [Collberg et al., 2015]. The academic literature on computational experiments has reported a lack of reproducibility, experiments results are briefly described and source code is seldom available, which makes it hard to utilise and extend the results [Freire et al., 2016].

nMANET provides technical information, an evaluation methodology, tools, and the code of a prototype. These technical details are available and can be tested and reproduced. The experiments include a description of methodology and conditions used to obtain the results. nMANET provides a compilation of the collected data sets, the scripts used for their analysis and the results, which were utilised in this thesis. In summary, nMANET provides an open repository of the source code[4], a description of the cases of each experiment, and a open repository of the collected experimental data sets, their analysis and data processing tools[5] [6]. Utilising these

[4]https://bitbucket.org/percyperezd/jnfdmanet
[5]https://bitbucket.org/percyperezd/jnfdexperiments
[6]https://sites.google.com/site/meshawaremobilenetworking/

tools, input data sets, and the evaluation methodology, it is possible to independently reproduce nMANET tests and experiments.

5.1.1.3 Scalability

Scalability in MANETs means the capability to operate with an increasing number of mobile nodes or increased interaction between them [Murthy and Manoj, 2004]. Implementations of nMANET were tested in real platforms such as Android and Linux-based platforms where the number of nodes was small. There are limitations to the size of a physical MANET that can be used for experiments. Apart from the cost of acquiring a larger number of devices, it is also difficult if not impossible to create reproducible experiments. Consequently, to scale up the tests and experiments, I utilise Mininet-WiFi[7], a simulator based on software-defined networking features in the Linux kernel. To my knowledge, at the time of writing, no other simulator exists that allows experiments with actual code and that simulates wireless communication.

Mininet-WiFi in a given version is deployed into a virtual machine to ensure that tests and experiments are repeatable, independently of the host system. Section 5.3 describes Mini-JNFD, which creates these virtual machines and controls the running of tests and experiments. Mini-JNFD also ensures the availability of the trace files that underlie the mobility of nodes (cf. Secion 5.3.1.2).

Given that JNFD is being developed for emergency applications, the size of the MANET will be non-trivial but also not very large - certainly not compared to the ambitions of the NDN project to replace IP as the basis for a future Internet. Practically, the size of the simulations I was able to run was determined by the number of usable GPS traces. The compute resources required to run larger simulations were therefore not an issue. For larger trace datasets, of course, larger computers would be needed. All experiments were run on a Dell XPS13 with a dual-core Intel i5 mobile CPU and 8GB of RAM.

5.1.2 JNFD Context and interactions

This section presents the relationship between JNFD and its external environment, which intends to delimit the boundaries and interactions of a JNFD forwarder.

5.1.2.1 JNFD over IP

NDN is functional over any type of transportation layer capable to forward datagrams, which means that NDN is functional over IP [Zhang et al., 2014]. While mappings to lower-level

[7] https://github.com/intrig-unicamp/mininet-wifi

protocols are being developed, I have followed the approach of using IP rather than trying to replace it. This also makes sense in the targetted Android environment where interfaces for sending raw Ethernet frames are not available, at least not from the Java runtime environment. The current implementation can use either UDP or TCP and is therefore compatible with the existing NDN codebase. However, a consequence of this choice is that JNFD inherits the IP address assignment problem. Note, that the use of IP does not mean that JNFD requires an IP routing infrastructure are communication is hop-by-hop only.

Deploying JNFD over IP represent a way to facilitate gradual integration into today's IP infrastructure and with NDN implementations. Similarly to Cisco[8] [9], JNFD uses IP until a clear path to adoption other low-level protocols exists. JNDN insulates the applications from the underlying transport, so making a transition should be relatively simple. To solve the IP address assignment problem, JNFD currently uses randomly assigned IP addresses (which only need to be unique amongst a group of neighbours) but it is also possible to use IPv6 link-local addresses generated from the network card's MAC address.

5.1.3 JNFD Architectural design

Figure 5.1 depicts the overall architectural model [Sommerville, 2010, p.147] that describes how components in JNFD are organised to satisfy the functional and non-functional requirements and how they communicate with each other. The following list briefly describes each component and its interaction with external nodes such as other forwarders, producers and consumers.

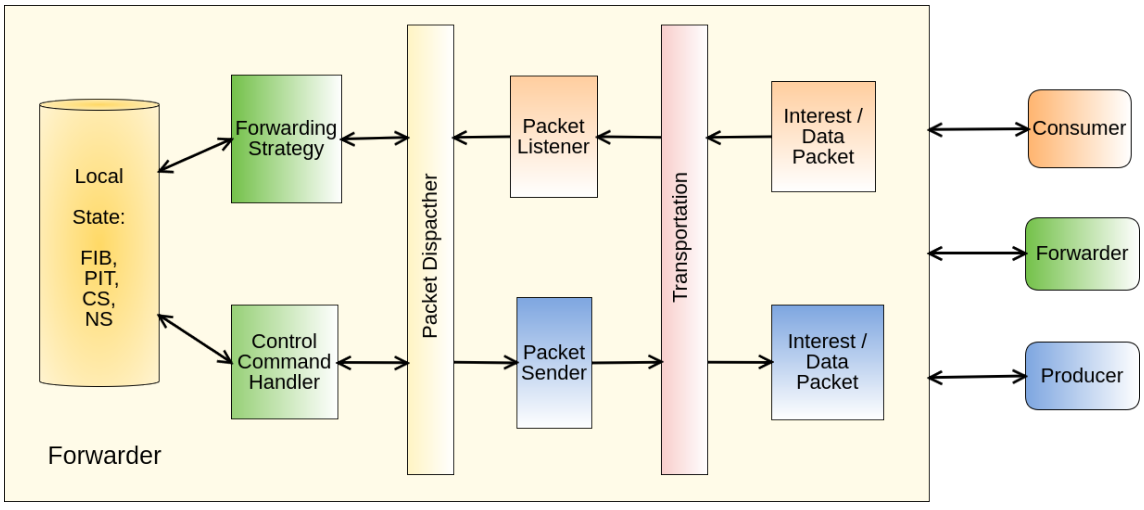


Figure 5.1: JNFD architectural model.

[8] https://wiki.fd.io/view/Cicn

[9] http://blogs.cisco.com/sp/cisco-announces-important-steps-toward-adoption-of-information-centric-networking

- **Interest and data packets:** These are the two main packet types that a JNFD forwarder should process. They are received from consumers that are interested in the content with a specific name, from producers who generate and store a specific named content as well as from other forwarders who pass on interests and data.

- **Transportation:** This component represents the type of transportation that carries the packets and is utilised by the external nodes such as consumers, producers and other forwarders. As JNFD aims to be independent of the type of transportation, this component provides an abstraction over the specific protocols used. At the moment TCP and UDP transport protocols are supported.

- **Packet listener and sender:** These two components are the main internal gates to JNFD that are responsible for the receiving and sending of packets via the configured protocol.

- **Packet dispatcher:** This component is one of the main parts of JNFD, as it dispatches tasks to the correct parts of JNFD to handle the request received or to reply with the information locally cached.

- **Forwarding strategies:** The logic to decide where to redirect a request that cannot be satisfied by the JNFD forwarder is executed in this component. It also replies with the requested information in the case that the local cache can satisfy the request.

- **Control command handler:** This component manages the control messaging between nodes. These messages are used to exchange up-to-date information between nearby nodes as well as being used by the forwarding strategy component to make forwarding decisions.

- **Local state:** These components store available information of the local and remote JNFD forwarders, producers and consumers. They are constantly updated by the forwarding strategy and control command handler components in order to have up-to-date information for further decisions. The local state breaks down into four tables: FIB, PIT, CS and NS.

5.1.4 JNFD Object oriented class model

Based on the requirements and the overall architectural model of JNFD, this section presents a summary of the implementation of JNFD in Java. This includes the main classes and packages implemented in JNFD.

5.1.4.1 Main JNFD modules and packages

JNFD was designed to be functional on both Android-based mobile devices and Linux-based devices such as desktop computers and laptops. Table 5.4 presents the three main modules of JNFD and their relationships. These three modules are built as independent .jar files. The module named `Jnfd` is the core module, which is the base for both platforms. It is the largest module as other modules contain only functionality specific to each runtime environment. The set of tests for continuous integration are located in `JnfdMain` along with the code that is specific to Linux (see also Section 5.4).

Module	Description
Jnfd	Contains the generic Java core for both Android and Linux-based implementations.
JnfdAndroid	Contains all implementation related to Android development.
JnfdMain	Contains all implementations related to Linux based command line version for desktop or laptops. It also contains implementations for continuous testing, which are presented in Section 5.4.

Table 5.4: Main JNFD modules

As both the `JnfdAndroid` and `JnfdMain` modules are strongly dependent on the `Jnfd` module, Table 5.5 lists the main packages designed in the core module of JNFD.

5.1.4.2 Main JNFD classes

This section presents more details of how JNFD was designed at class level by presenting the UML class diagrams of Figure 5.2, Figure 5.3 and Figure 5.4. The combination of these three figures represents the static view of JNFD, which visualises the relationships between the main classes of JNFD.

As presented in Table 5.5, JNFD utilises a builder design pattern at booting time. It creates all necessary instances for JNFD, which includes transport, listener & sender for nodes status, strategy and the`PacketDispatcher`. Once the builder finishes to create all objects required by JNFD, the forwarder is ready to receive requests from nearby nodes. In the case that an incoming interest or data packet arrives through the transport class, this request is handed to a callback in the packet dispatcher. The packet dispatcher is the instance that starts to handle incoming packets by following the flow diagrams of Figure 4.2 for interest packets, and Figure 4.4 for data packets. These two flow diagrams are the main reason to link the packet dispatcher to the

Package	Description
`jnfd` (default)	Contains the main jnfd classes that start all instances and create the relations between objects. It applies dependency injection with the builder design pattern to create objects at booting time. This ensures that the components show in the architectural model of the forwarder can be replaced without requiring changes to other components (cf. Figure 5.1).
`network`	Contains helper class for creating IPv6 link-local addresses and for managing Wi-Fi configurations.
`os`	Contains the implementations that manage ad-hoc communication, which includes starting and stopping ad-hoc mode, getting wireless interface status, and testing connectivity.
`status`	Contains the implementations for node status tables, listeners and senders.
`strategy`	Contains the implementations of the forwarding strategies of JNFD, which were presented in Section 4.2.3 and the additional features of JNFD presented in Section 4.7. This package is the place where new JNFD strategies can be implemented as further detailed in Section 5.7.
`tables`	Contains the implementation of the main JNFD tables: PIT, FIB, CS and NS. This is the place allocated for new tables for new types of forwarding strategies or new functionalities in JNFD.
`transport`	Contains the implementation to receive packets from any in-range node, and also send packets to neighbours. As explained in Section 5.1.2, JNFD needs to integrate the current NDN implementation. Consequently, this package implements transport types such as TCP and UDP.
`util`	Implementations in this package can be classified as common functionality as well as constants such as identifiers and default thresholds.
`proto`	Contains the message specification used for JNFD to exchange messages between nodes, in particular to exchange content between nodes through node status packets.

Table 5.5: Lists of packages of the core jnfd" module

strategy, the nodes status listeners and sender and the transport classes. These relationships are also depicted in Figure 5.2.

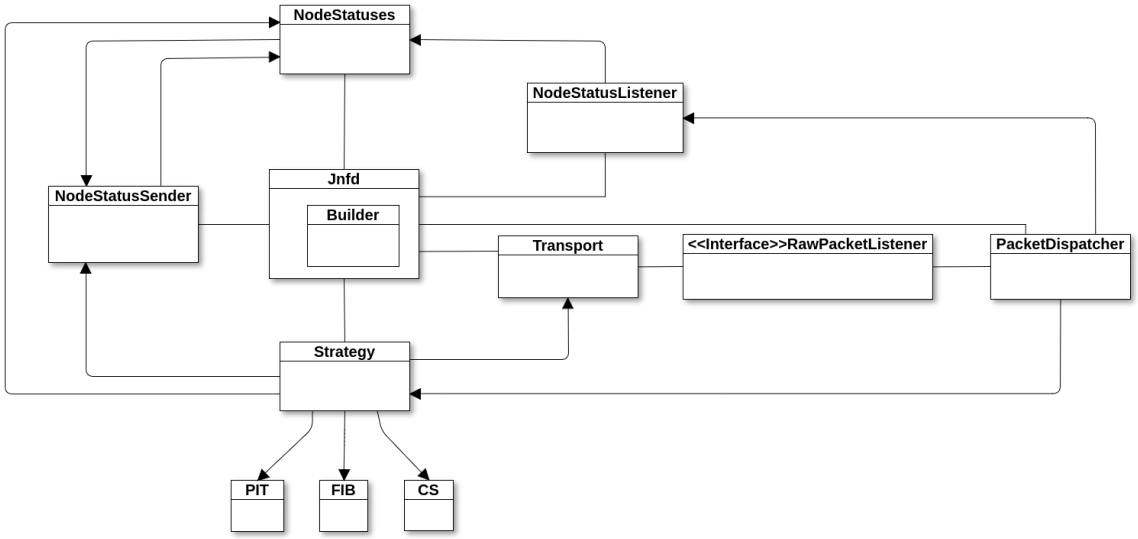


Figure 5.2: UML class diagram for jnfd class.

In the case that an interest packet cannot be satisfied by local caches, JNFD starts the forwarding strategy process as explained in Figure 4.2 and 4.3. Figure 5.3 depicts the class diagram for this forwarding strategy process, and the `AbstractStrategy` class represents the core of the forwarding strategy implementation. This class is extended by the currently available forwarding strategies and it should be extended by new forwarding strategies in JNFD. The `NodeStatusListener` and `NodeStatusSender` classes are also included in the relationships as they help to exchange status information between nodes, in particular in the case JNFD starts to execute the prefix discovery and JNFD additional features as explained in Section 4.6 and Section 4.7, respectively.

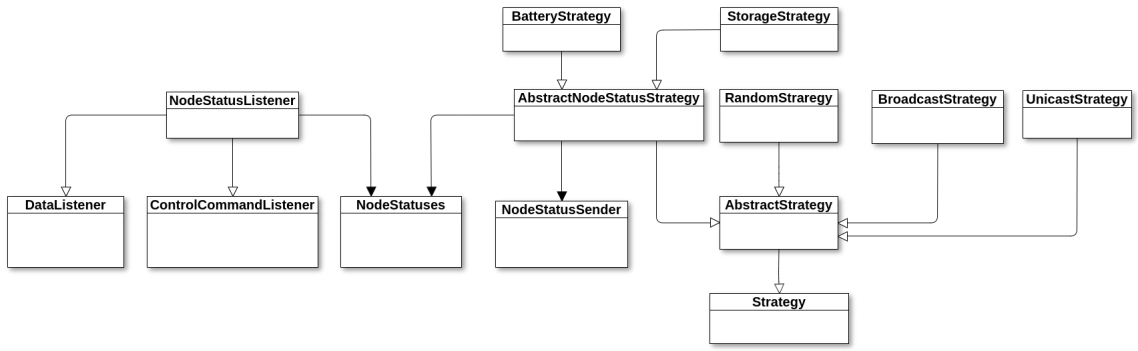


Figure 5.3: UML class diagram for strategy class.

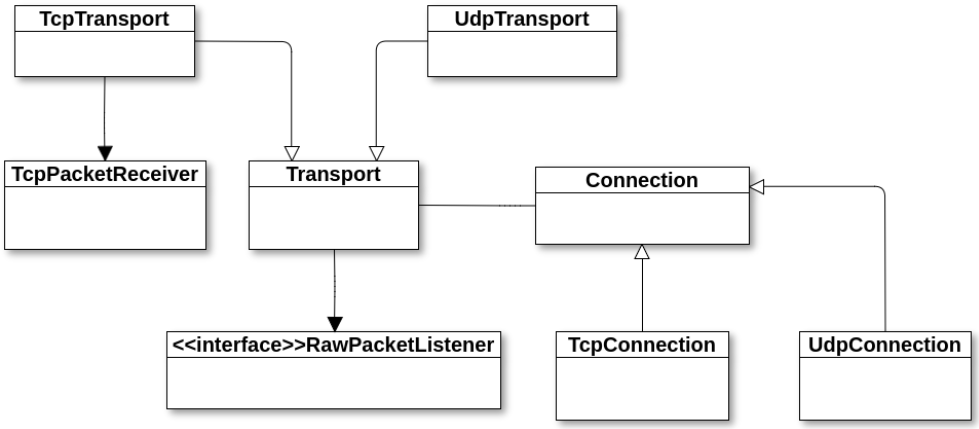


Figure 5.4: UML class diagram fro transport class.

Finally, the `Transport` class depicted in Figure 5.4 describes the relationships between type of transportation and type of connection. A `Connection` contains the information necessary to communicate with another node, such as a TCP or UDP Socket. JNFD is designed to support TCP and UDP connections, which makes JNFD compatible with the current NDN platform implementation. However, if a new type of transport protocol is implemented, such as raw Ethernet sockets, a new implementation of the `Connection` and `Transport` interfaces needs to be implemented. In the case a packet arrives at the forwarder, the class that implements the `RawPacketListener` interface buffers the packet for further processing, as illustrated in the class diagram in Figure 5.2.

Naming in the FIB The FIB stores information about possible next hops for a given name prefix. The data in it can be obtained through two mechanisms. The first is through prefix discovery, described in Section 4.6, which leads to entries that contain the specific name that was contained in an interest. While the information obtained this way is correct, this process potentially leads to a large number of entries in the FIB. It would be highly desirable to be able to collapse entries with a common prefix if they are serviced by the same producer and should lead to the same next hop selection. This was discussed in Section 4.7.2. Unfortunately, without any information about the actual prefix registration, collapsing the entries may produce incorrect results.

Another way that data about possible next hops can be obtained is through prefix propagation, discussed in Section 4.7.4. This method has the advantage that it populates the FIB with prefixes rather than observed names. Currently, prefix propagation is implemented by means of broadcasts triggered by the producer registering with a forwarder. Prefix propagation is an optional feature that can be activated on the forwarder. If it is turned off then the network relies prefix discovery

via node statuses to identify possible next hops.

5.2 JNFD implementation process

This section presents all relevant activities involved in the software development of JNFD, which include the following topics: reuse of existing software artefacts, configuration and build management utilised during the development process as well as the host targets where JNFD was tested. All these parts are presented in the following paragraphs.

5.2.1 Software reuse

JNFD is implemented re-using appropriate existing components, systems and code. For example, it reuses the JNDN client libraries provided by the NDN project [10], the proto3 language for specifying message formats provided by Google [11], as well as the existing Java libraries provided by Android or Linux based platforms. JNFD reuses code and concepts at the abstraction and object level rather than at component and system level, as presented in the following paragraphs.

Abstraction level At this level, JNFD utilises creational design patterns such as the builder and singleton patterns [Gamma et al., 1994]. The strategy pattern is used in the forwarding strategies, which are selectable at runtime and can potentially be changed depending on circumstances other than user selection.

Class/object level As initially mentioned, JNFD reuses classes and objects from exiting libraries, such as objects from JNDN, proto3 from Google, Spock for testing, and Android libraries. Table 5.6 lists the relevant objects that the JNFD code reuses from existing libraries.

Component level At the component level, JNFD adapts and extends a set of objects mainly for interfaces that have direct contact with the user, such as the case of Android libraries used to collect and display information to the screen of the mobile device.

System level JNFD does not reuse entire applications or software for its implementation. One exception to this is the `ip` command line tool that allows to change Wi-Fi network interfaces to ad hoc mode. In the JNFD development process, existing NDN applications as well as IP-based applications such as `curl` were used for system testing.

[10] https://github.com/named-data/jndn
[11] https://developers.google.com/protocol-buffers/docs/proto3

Library Name	Brief description
Java concurrent hash maps	All internal tables in JNFD are implemented using concurrent hashmaps. For example, in the case of the PIT, the key of the hashmap is the interest packet and its value is a set of connections to nodes that have requested a content of a particular name.
Google protocol buffer messages	JNFD utilises the features of proto3 from Google[12] to create its own data structure of messages to be exchanged between nodes. It lets proto3 manage the details of reading and writing units of messages. JNFD utilises the encoding and parsing functionalities of Proto3 by using the generated getters and setters.
Apache commons daemons	JNFD utilises JSVC[13] from the Apache common daemon[14] libraries to run JNFD as service on Unix-based platforms. By using JSVC to run JNFD as a daemon, JNFD reuses the methods provided by JSVC to start, stop, destroy, and initialise JNFD.
JNDN client libraries	As JNFD needs to be compatible with NDN applications, JNFD reuses objects provided by JNDN, such as the Name, Interest, Data, Blob, KeyChain, and MetaInfo classes.
Java Runtime	JNFD utilises the Java Runtime's functionality to execute Unix shell command lines at a running time, for example, to enable Ad-hoc mode on Linux-based laptops.
Java NIO networking	JNFD implements sockets using the non-blocking IO and multiplexing functionalities of the Java NIO libraries. It mainly utilises socket channels and selectors for TCP and UDP connections.
Android libraries	The JNFD module for Android devices utilises the libraries provided by Google for developers[15].
Java logging	JNFD uses the standard Java logging libraries to generate logs under the JNFD format that is discussed in Appendix 5.6.
Args4j command line parsing	JNFD uses the Java classes of the Args4j[16] library to parse command line arguments provided on Linux systems.

Table 5.6: List of relevant third-party libraries used by JNFD.

5.2.2 Configuration management

Changes in JNFD implementation during its development were managed by the following activities:

Version management: JNFD source code was initially created in Bitbucket repositories using the Git[17] version control system for tracking code changes. JNFD also includes unit, component and system testing code in its Git repository.

System integration: The JNFD Git repository includes the scripts and procedures for installation. As JNFD is open source, it provides the necessary instructions for system integration with other platforms.

5.2.3 Host target development

JNFD software was developed using Android Studio[18] for both the Android and the Linux version. A Gradle[19] build process allows the system to be built from the command line. Tests are also executed from the command line as are the experiments and the analysis of the collected experimental data.

JNFD was tested on Android smart-phones such as the Motorola Moto E and Moto G as well as the Samsung Note 4 and Galaxy. JNFD can be deployed in Android smart-phones by installing the .apk file through Android Studio, or through the `adb`[20] command line. Additionally, it was tested on Linux-based systems. For Linux systems, JNFD can be deployed by shell scripts, which compile and create the links to the binaries of JNFD.

5.3 Mini-JNFD: Scalability and reproducibility of JNFD

The Mini-JNFD testing framework for JNFD testing and experimentation is based on Mininet-WiFi. The motivation to create the Mini-JNFD testing framework is the lack of a tool to test JNFD on a small and medium number of mobile nodes connected in ad hoc mode, as well as under controlled conditions using models for the movement of nodes. This section summarises how Mininet-WiFi was utilised by Mini-JNFD to accomplish this goal.

[17] https://git-scm.com/
[18] https://developer.android.com/studio/index.html
[19] https://gradle.org/
[20] https://developer.android.com/studio/command-line/adb.html

Figure 5.5 presents the main components of the Mini-JNFD testing framework. The host machine executes Android Studio as the development tool of the source code of JNFD, which is linked to a Git remote repository located in Bitbucket[21]. Additionally, the host machine executes a Virtual Machine, in which Mininet-WiFi is installed. The Mini-JNFD testing framework relies on the Testing and Experiment Manager (TEM), which links all previous components to a repository of experiments and results.

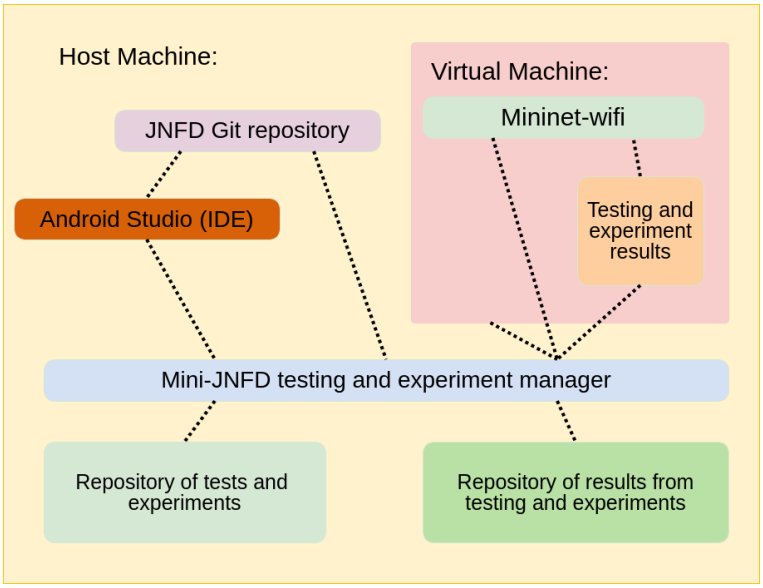


Figure 5.5: Mini-JNFD general component diagram, Testing and Experiment Manager (TEM).

TEM in Mini-JNFD consists of a series of Unix shell and Python scripts that manage and execute the tests and experiments in Mininet-WiFi in the Virtual Machine. TEM initially verifies whether there are pending commits that need to be pushed to the Git repository. Otherwise, it compiles the source code automatically to get the JNFD jar files, which are tested in Mininet-WiFi.

TEM copies tests, experiments and the latest compiled JNFD jar files to the Virtual Machine. This is so that Mini-JNFD executes the tests and experiments with the last up-to-date modifications on the JNFD source code. Once TEM is logged in to the Virtual Machine, it executes the tests or experiments in Mininet-WiFi and collects the results in an XUNIT test case file using an XML format. TEM executes all experiments or tests sequentially and generates individual XUNIT test cases per test or experiment.

Finally, TEM aggregates all results from the XUNIT test cases and transfers them to the host machine. The aggregated file contains a summary of all tests or experiments including the details of each of them. TEM generates a new folder every time the experiment or test is repeated, which

[21] www.bitbucket.org

makes it possible to reproduce tests or experiments as many times as required. Note that TEM tags each folder with the date and time at the beginning of the execution. In this way, each test or experiments can be debugged manually by reading the content of the generated logs and results.

Mini-JNFD allows the JNFD daemon to be tested on a small to medium number of nodes by using the Mininet-WiFi[22] wireless network simulator. The main motivation for the creation of Mini-JNFD is to offer a place for testing and experimentation with more than a few mobile nodes. The financial limitations to acquire more than 10 smart-phones are solved by Mini-JNFD as is the problem of creating repeatable experiments. The following paragraphs describe the internals of Mini-JNFD and how testing and experimentation can be scaled, reproduced and simulated by any researcher or developer interested in enhancing JNFD.

5.3.1 JNFD Scalability

Mini-JNFD utilises the available functionalities of Mininet-WiFi to build an Ad-hoc network inside Mininet[23]. Mini-JNFD installs in a virtual machine all the modules from Mininet-WiFi in order to simulate different network topologies. Mini-JNFD provides its own Python scripts that create mobile stations and link them in ad-hoc mode to form a MANET. It utilises the plot graph features that Mininet-WiFi has to display the topology of the MANET and the mobility of nodes.

As described in the following paragraphs, the mobility of nodes in Mini-JNFD can use one of two main approaches: one is utilising the available mobility models of Mininet-WiFi, and the second approach is utilising real GPS traces such as those collected by the Crawdad project[24].

5.3.1.1 Mobility models

The available mobility models of Mininet-WiFi include Random Walk, Random Direction, Random Way Point, Gauss Markov, Reference Point,Time Variant Community and Truncated Levy Walk. From this list of models, I consider those that based either on empirical work or on theories about mobility.

Time-variant Community Mobility Model: This model assumes that mobile nodes periodically visit communities and that they also create periodical re-appearance on the same locations. It assumes that communities have a reference point that mobile nodes aggregate around. These community references are not static but instead follow a random direction model. Additionally,

[22]https://github.com/intrig-unicamp/mininet-wifi
[23]http://mininet.org/
[24]http://crawdad.org

the level of aggregation controls the proximity of mobile nodes to the references point of the community [Hsu et al., 2007].

Reference Point Group Mobility Model: This model is based on the assumption that group motion happens frequently in MANETs, which makes it possible to partition the network into groups. According to this model, the trajectories of these groups follows a random direction, and mobile nodes follow a random walk around the group centre. Additionally, as mobility models are used for mobility tracking, this model includes the parameter 'aggregation', which controls how close the mobile nodes are to the group centres [Hong et al., 1999].

Gauss-Markov Mobility Model: This model initially assigns a current speed and direction to each mobile node. Speed and direction are later updated with new values at fixed intervals of time. Therefore the movement of mobile devices is achieved. However, the new values for speed and direction are calculated based on the current values and two random variables from a Gaussian distribution. The randomness of these two variables is controlled by a third variable, which values vary between 0 and 1 [Camp Tracy and Vanessa, 2002].

Truncated Levy Walk: The Truncated Levy Walk Model [Rhee et al., 2011] is based on experimental data consisting of one thousand hours of GPS traces which involves 44 outdoor volunteers in two different campuses, a metropolitan area, a theme part, and a state fair. Based on this empirical data, this stochastic model generates patterns of human walks. This is the only model built into MiniNet-Wi-Fi that is based on empirical data.

Condidering the above models, the Truncated Levy Walk seems to be the most appropriate to choose as a basis for JNFD testing and evaluation. There is, however, an additional alternative, which is to use GPS traces directly, without an intervening stochastic model. This is discussed in the following.

5.3.1.2 Mobility traces

MiniNet-Wi-Fi can also utilise real GPS trace files to determine the behaviour of the mobile nodes, which includes trace files of Cartesian coordinates. In the following, I am using a dataset[25] generated through tracing 41 laptops moving on a sports field [Gray et al., 2004], provided by Gray et al. through a repository run by the Crawdad project[26].

The GPS trace file contains complete data for 29 nodes out of 41, and their GPS coordinates include a time stamp, latitude, longitude and altitude values. As Mininet-WiFi does not read GPS

[25] https://crawdad.org/dartmouth/outdoor/20061106/
[26] http://crawdad.org

coordinates, I converted these GPS traces into projected coordinates (Cartesian coordinates). The methodology to plot the mobility traces was based on finding a relative reference after the conversion, these relative values are the result of subtracting the minimum value. A second modification of the data set provided by Crawdad project is the modification of the time stamp. As Mininet-WiFi only recognises a sequence of numbers, Mini-JNFD converts the time stamp to sequence of numbers by arithmetic subtraction with respect to the minimum timestamp of the trace file. Consequently, the sequence number represents the amount of time in seconds that the nodes need to wait until they move to the next position. I provide open access to the scripts that make these conversions and the results can be found in the repository of experiments of JNFD[27]. These results show that not all trace files are suitable to be utilised, and the Listing 5.1 explains the detailed reasons.

```
Trace File Node -->  Inconsistency found
=========================================
11 --> The original file has errors at line 1521
16 --> The timestamp is negative
18 --> The original file has errors at line 4
30 --> The original file has errors at line 2197
35 --> The original file has errors at line 3613
38 --> The original file has a 0.000000 coordinate at line 2060
44 --> The original file has a timestamp negative
49 --> The original file has a gap in the time between line 1 and 2: "1065916800.000 -
1066400799.000" = -483999.000
```

Listing 5.1: Trace files of nodes with errors

Finally, the total number of correct trace files available for use is 29, and are listed in Listing 5.2

```
crawdad.12.seq  crawdad.1.seq   crawdad.32.seq  crawdad.3.seq   crawdad.50.seq
crawdad.13.seq  crawdad.21.seq  crawdad.33.seq  crawdad.40.seq  crawdad.7.seq
crawdad.14.seq  crawdad.25.seq  crawdad.34.seq  crawdad.41.seq  crawdad.8.seq
crawdad.15.seq  crawdad.26.seq  crawdad.36.seq  crawdad.45.seq  crawdad.9.seq
crawdad.17.seq  crawdad.28.seq  crawdad.38.seq  crawdad.47.seq
crawdad.19.seq  crawdad.31.seq  crawdad.39.seq  crawdad.48.seq
```

Listing 5.2: List of trace files converted without inconsistencies: crawdad.<NODE>.seq

5.3.2 JNFD Reproducibility

Reproducibility is another important feature of Mini-JNFD. Mini-JNFD offers the option to build the necessary infrastructure by providing all scripts and tools, which are described in the following paragraphs.

[27] https://bitbucket.org/percyperezd/jnfdexperiments

Vagrant Mini-JNFD utilises open source software, such as Vagrant[28], to provision the runtime environment in which tests and experiments are executed and thereby supports reproducibility of the environment and infrastructure. The provisioning scripts are part of the JNFD source code and can be used to set up the environment for Mini-JNFD in minutes.

Automation Mini-JNFD includes the necessary tools to re-execute tests from a command line, therefore testing and experimentation can be reproduced by anyone interested in repeating the actual functionality of JNFD or testing newly developed features. By mimicking the existing Mini-JNFD tests or experiments, new features can be tested without too much effort using the Testing and Experimentation Manager (TEM), which is utilised to execute tests and experiments from the command line.

5.4 JNFD Continuous integration

JNFD and Mini-JNFD follow the practice of continuous integration [Humble and Farley, 2010] by building and testing JNFD on every push to the Git repository. The process of building and testing for JNFD is automated through the Mini-JNFD tools.

Tests of JNFD are distributed across three main groups, which represent three levels of granularity: unit-, component- and system testing. Section 5.4.1 presents the unit test cases used for Test Driven Development (TDD). Component testing is presented in Section 5.4.2. JNFD uses Spock[29] for unit- and component testing. Spock was chosen for its functionality to express human-readable tests specifications and its compatibility with most IDEs.

Finally, system testing of JNFD is mainly based on Mininet-WiFi and the Testing and Experiment Manager and it is divided into two parts. The first part includes testing JNFD on real Android mobile devices such as Moto E smart-phones. However, due to the costs to acquire a large number of mobile devices, I use Mini-JNFD to simulate a larger number of mobile devices linked in ad-hoc mode.

5.4.1 Unit testing

Table 5.7 lists the main individual tests for objects classes of JNFD. This testing is focused on the functionality of classes and methods of the code of JNFD. For more technical details, the source code of JNFD provides the definitions of each unit test.

[28] https://www.vagrantup.com/
[29] http://spockframework.org/

Unit test Name	Brief description
Network	It tests Wi-Fi settings of a JNFD forwarder, for example: channel 3 should be map to 2.422 GHz
Operating System	Tests whether command line operations through the `JavaRunTime` class.
Node status	Tests the functionalities of the JNFD node status features. For example methods from NodeStatus, NodeStatusSender and NodeStatusListener classes.
Forwarding Strategies	Testing for the behaviour of a JNFD forwarder when it uses a specific forwarding strategy, for example: Random Strategy.
Internal tables	Testing behaviour of the PIT, FIB and CS tables of a JNFD forwarder.
General	Testing of main classes of JNFD, such as the PacketDispatcher class.

Table 5.7: List of relevant unit tests of JNFD.

5.4.2 Component testing

Component testing in JNFD covers tests of the relevant modules of JNFD and its interactions, such as granularity in FIB. Table 5.8 provides the relevant tests for a JNFD forwarder. The source code of JNFD provides the more technical details and it helps as a reference for further component tests.

Component test Name	Brief description
Granularity in FIB	Tests how JNFD can reduce the number of prefixes in order to minimise the number of FIB prefixes to be exchanged between forwarders. This test includes the reverse name component algorithm presented in Section 4.7.2.
Life time of interest packets	Tests that interest packets should be removed from the PIT tables after their life time expires.
Registration of a prefix in FIB	This test verifies whether the prefix is updated in the FIB in the case that a producer registers a new prefix.
Serving content from cache	This test verifies whether a forwarder satisfies a request of a particular name, in the case the content was previously cached in CS.
Updating a PIT entry	This test checks how JNFD updates the PIT table in the case that more than one interest packet with the same name is requested of a forwarder.

Table 5.8: List of component tests in JNFD using Spock and Mini-JNFD.

System test	Brief description
Prefix registration of producers	To test whether a forwarder correctly updates the FIB table with the prefix sent by the producer.
Retrieving local cached content	To test whether a forwarder replies requests by using the content cached in the CS.
Interest name retained by PIT	Tbasico test whether a forwarder does do nothing in the case that the incoming name of interest is already in the PIT.
Forwarding interest using FIB next hop selection	To test whether a forwarder redirects the interest packet to the most suitable next hop, where the most suitable next-hop is selected by the forwarding strategy module called: FIB next-hop selection.
Forwarding interest by flooding	To test whether a forwarder updates its local FIB and node status table with information from one-hop nodes. Also this test whether the interest packet is broadcasted in the case that non of the neighbours contain the prefix of interest.
Testing connectivity	To test basic communication between any two nodes using the traditional ping.

Table 5.9: List of main JNFD tests using Android smart-phones.

5.4.3 System testing

JNFD system testing is divided into two principal sub-groups: testing in Android smart-phones, and testing in a network simulator through the Mini-JNFD framework. Both sub-groups are summarised in Table 5.9 and Table 5.10, respectively.

System testing in Android smart-phones: In the case of testing in Android smart-phone (Table 5.9), the functionality of JNFD was tested based on the available number of mobile devices to form an Ad-hoc network scenario consisting of one consumer, two producers and one forwarder. In this configuration, the consumer requests the content of a name of interest from a forwarder that makes a forwarding decision between the two producers in order to retrieve the content of the name of interest.

To verify basic connectivity in Linux based devices, the standard ping command is used between members of the MANET. In Android devices, in case that the standard Android Debug Bridge (ADB) is not available to execute ping commands from inside the smart-phone, JNFD offers an option to run the ping command from an Android application by calling a Java wrapper class.

System testing in Mini-JNFD: Table 5.10 lists the main tests of JNFD functionalities in Mini-JNFD. These tests are available in the source code of JNFD in order to reproduce them through the Mini-JNFD framework.

System test Name	Brief description
Broadcast Interest Packets	It tests the JNFD feature presented in Section 4.7.1, that JNFD can control the broadcast of interest packets by limiting the number of broadcasts and delaying them across the network.
Producers prefix propagation	It tests how a forwarder propagates the prefix registration requested by a producer. This JNFD feature is expanded in Section 4.7.4.
FIB-next hop selection	These tests verify whether a forwarder selects the correct next hop or hops according to the type of strategy set at booting time, such as random and unicast strategies. Section 4.5 provides more references.
Reverse Path Updating through FIB	This test, presented in Section 4.6, shows whether JNFD populates the correct FIB entry in the case that intermediate forwarder re-broadcast an unsatisfied interest packet until reaching the producer.
Prefix Discovery	This test verifies whether a forwarder starts the prefix discovery process in the case that the name of interest cannot be satisfied. This principal feature was described in Section 4.6.
Initial FIB configuration at booting time	This test is for new JNFD forwarders who join a MANET. As it was described in Section 4.7.3, JNFD updates the FIB of new forwarders based on the available prefixes from nearby nodes.
Minimal scenario to retrieve content	This test verifies whether a consumer can retrieve the content from a producer through one forwarder.

Table 5.10: List of main JNFD system tests using Mini-JNFD.

5.5 JNFD Continuous Delivery

JNFD and Mini-JNFD follow the following principles of continuous delivery [Humble and Farley, 2010], in particular, to pipeline all modifications in a way that the effect of a modification in the source code can be detected by Mini-JNFD testing and experimentation framework. In the case that the new modification passes all the Mini-JNFD tests, it represents a higher level of confidence that JNFD will be functional as expected and the modifications can be released. The process of continuous delivery consists of the following aspects:

Automate as much as possible The process of building, deploying, testing and releasing JNFD software is visible to everybody, in order to create collaboration and feedback. Therefore, bugs and problems can be identified. To achieve this, Mini-JNFD automates the testing and experimentation process during the development and build process. Consequently, JNFD tests and experiments are repeatable and software development reliable.

Simple and easy testing Building and testing of JNFD are automated in Mini-JNFD through Gradle and the TEM tool. This tool checks for the last update modifications on JNFD, compiles them to automatically generate the respective jar files and release them in the Virtual Machines where Mininet-WiFi is installed. All this pipeline is centralised in one simple command line tool called TEM, as described in Figure 5.5.

Version control JNFD and Mini-JNFD manage code changes through Git version control. As the hardware infrastructure cannot be in version control, Mini-JNFD utilises virtualisation through Vagrant and Virtual Box tools. The configuration files, scripts and procedures all are in a Git repository. JNFD and Mini-JNFD keep all information in three main git repositories: the source repository, the experimental data repository and the Mini-JNFD repository. This information includes source code, test, experiments, continuous integration scripts, virtualisation configurations, data collected from experiments, and technical references.

Dependencies JNFD and Mini-JNFD utilise external artefacts such as JNDN and Mininet-WiFi. These dependencies are managed through configuration scripts in groovy, gradle tasks and shell scripts.

5.6 JNFD Logging implementation structure

JNFD logs information to files using the Java Logging library in order to provide support for debugging, testing and experimentation. JNFD logs have the following structure of components, where the main delimiters is a comma:

```
Date and Time, JNFD profile, Log type, Java Class Name, Java Method Name, JNFD message
```

In this structure JNFD profile could be CONSUMER, PRODUCER, FORWARDER or some other name that identifies the node itself. The Log type identifies the category of the log, which can be INFO, DEBUG, or WARNING. The Java class and method names corresponds to the names of the Java classes and methods in the source code where this message is located. Finally,

the JNFD message part contains information about the internals of JNFD and it has the following structure that is also separated by comma.

```
Action, Object type, Result, Name, Additional Information, fromIp, fromPort, toIP, toPort
```

The action field refers to a verb that express a type of action executed by JNFD. The object type refers to the JNFD object that is affected by the action. The name corresponds to the name that is associated with the action. The fromIP and fromPort components correspond to the address of the node from where this transaction came, and the toIp and toPort are the components that correspond to the address this transaction goes towards. The result field indicates the status of the transaction or action. Finally, the additional information field is utilised by JNFD to add more information that helps to clarify the meaning of the transaction.

Listing 5.3 shows three examples of JNFD log to clarify the structure fo the components of each action and its description. Additionally, Table 5.11 lists all tags utilised by JNFD in a forwarder node.

```
# JNFD log in a forwarder when a prefix registration request is received from a producer.
26/10/2017
04:28:37.0214,FORWARDER,INFO,com.group.yali.jnfd.PacketDispatcher,handleRegistration,Received,
REGISTRATION,OK,/a/b/c,,10.0.0.5,59456

# JNFD log when a forwarder updates it internal FIB table with a new prefix.
26/10/2017
04:28:37.0214,FORWARDER,INFO,com.group.yali.jnfd.tables.Table,add,Updated,FIB,OK,/a/b/c,the
name is
appended to existing one,10.0.0.5,59456

# JNFD log in a forwarde when a forwarder sends a registration acknolegdement to the producer.
26/10/2017
04:28:37.0235,FORWARDER,INFO,com.group.yali.jnfd.PacketDispatcher,sendBackRegistrationAck,Sent
,REGISTRATIONACK,OK,/a/b/c,,,,10.0.0.5,59456
```

Listing 5.3: JNFD log examples

The logging function are used to investigate the behaviour of the nodes in the network. As they run on the same operating system kernel, their clocks are synchronised and it is possible to use timestamps to align events from different nodes with each other. This makes it possible to trace the ‘biography’ of an interest, from the time it is issued by the consumer to the time it hits the producer or a forwarder containing the data and on to the point where the interest is satisfied as the consumer has received all the data.

ACTION TAG	OBJECT TAG	RESULT TAG
Sent	INTEREST	SUCCESS
Received	DATA	FAIL
Selected	REGISTRATION	ERROR
Redirected	PIT	PENDING
Broadcast	FIB	AVAILABLE
ReBroadcasted	CS	UNAVAILABLE
NoBroadcast	PACKET	CANCEL
Nothing	PREFIXES	OK
Removed	PREFIX	RUNNING
Refused	GRANULARITY	
Accepted	REBROADCASTINGINTEREST	
Error	REGSITRATIONACK	
Got	NAME	
Inserted		
Grained		
Matched		
Cancelled		
Updated		
Update		
Checked		
Setting		
Propagated		
Repropagated		
LimitDelayBroadcast		
Delay		

Table 5.11: Naming of the tags for JNFD logging

Likewise, it is possible to investigate the behaviour of an individual node such as a forwarder over time or to generate aggregate statistics about nodes such as the number of interests received or the number of data packets sent.

In addition to debugging and JNFD error tracking, JNFD logs are used in component and system testing, as they are the patterns used to identify whether a transaction or action was successfully completed. This is because JNFD follows the Continuous Testing and Integration practices as detailed in Section 5.4 and 5.5 respectively.

Finally, the logs are used in the experiments, where they are used to measure the time between events such as the sending of an interest and the first data packet arriving as well as to count the number of events over time such as measuring the degree to which broadcasts are used for prefix discovery.

5.7 Process for writing new forwarding strategies

Table 5.12 describes the steps to implement a new strategy using JNFD and Mini-JNFD. A new forwarding strategy can be implemented in JNFD by following the existing implementations of JNFD forwarding strategies and by following the steps described in this section.

As the "Jnfd" module contains the code that is reused by other modules, adding a new forwarding strategy to JNFD should start from this module by adding a new Java class in the `strategy` package. This new forwarding strategy should extend the `AbstractStrategy`, which contains the methods that help to forward packets from the forwarder to other nearby nodes. This new forwarding strategy class should contain the logic to select the next hop, and as a result, it should invoke the transportation class to send the interest packet to the select next hop. As an example to follow, the code of the random forwarding strategy can be inspected for more technical details.

Artefact	**Implementation**
`strategy` package	Add the new strategy Java class that extends the `AbstractStrategy` class and override the `forwardPacket` method.
`transport` package	Invoke the transport classes to redirect the interest packet to the hops defined by the new forwarding strategy class.

Table 5.12: Procedure to implement a new strategy in JNFD

5.8 Chapter summary

This chapter presents the design, implementation, testing and evaluation of JNFD and Mini-JNFD framework, which represents the prototype of nMANET described in Chapter 4. JNFD design includes the main requirements, context and interactions suggested by the nMANET approach, which are the base for the architectural design and Object Oriented class model of JNFD. Additionally, presents the logging format that is used across JNFD modules, and that is heavily monitored by the continuous system testing. On the other side, JNFD implementation includes details of artefacts reusability such as abstractions, objects, components, and systems. It also represents the baseline to present scalability and reproducibility through the Mini-JNFD test and experimentation framework.

Finally, this chapter present JNFD and Mini-JNFD as software that follows the continuous integration practices and continuous delivery principles. It provides the instructions of how new forwarding strategies can be implemented within JNFD and Mini-JNFD, the limitations that JNFD faces at the time of this writing, and the statistical data analysis for evaluation.

CHAPTER SIX

nMANET FRAMEWORK VALIDATION

6.1 Introduction

This chapter presents a series of experiments that aim to validate the implementation of the nMANET approach, and the evaluation framework. The validation of JNFD as a framework includes a repetitive cycle starting from the design of the experiment and ending in the presentation of the expected results. The methodology used in each repetition follows three main steps: the first one consist of describing the design of the experiment and the expected outcome from data analysis of results. The second step consists of executing the experiment in the JNFD framework with the aim of explaining how this kind of experiment can be handled by this framework. Finally, the third step consists of collecting the experimental data for data analysis and visualization. The aim of the third step is to show that the experiment behaves as it was expected during the design and that the analysis of the collected data supports this conclusion.

Note that this chapter focuses on the validation of the JNFD framework through two main group of experiments. The first group focuses on network traffic measurements using simulations, while the second group collects energy consumption measurements from the batteries of real mobile devices, such as Android smartphones. These two groups of experiments include representative examples that researchers can use as a reference for further and new experiments.

The experiments presented in this chapter show that the proposed framework provides a place to research new NDN forwarding strategies in a MANET, and also allows an analysis of results through the use of logs. The design of experiments and the analysis of results, presented in this

chapter, are left to the designer, therefore the justification of each design is pushed out of the framework, this also includes the strategy used for data analysis.

Finally,this chapter shows that the framework comes with a number of forwarding strategies. As forwarding strategies are not the focus of this thesis, these strategies represent a relevant set of examples that can be build upon in future work. All these experiments focus on enabling more responsible research on applying NDN in MANETs through reproducibility.

6.1.1 Methodology for experimental research

This chapter presents the experiments used to validate the nMANET framework. In each experiment, a *workload* is applied to the *system under test* under well-specified conditions determined by parameters for the system, the workload and the environment [Effelsberg et al., 2013], as illustrated in Figure 6.1.

This methodology follows the basic techniques of experimental research, where independent variables can be systematically controlled and all other variables can be held as constant. Thus, the experiments progressively produce more confident results when any changes occur in the dependent variable [Marczyk et al., 2005].

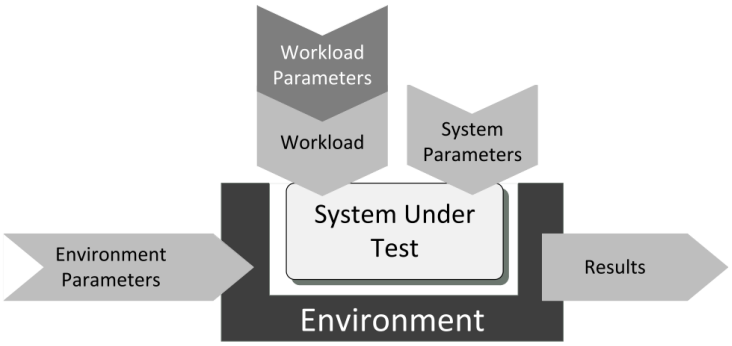


Figure 6.1: General benchmark model for MANETs [Effelsberg et al., 2013]

The system under test and its parameters are executed in an environment that provides the resources necessary to make the system functional. Additionally, as nodes of a MANET usually make autonomous decisions and manage their own resources without the presence of an authority supervising their behaviour in the network, a workload should be applied to each member individually. Finally, the parameters of the tested system are measured by quantifying metrics.

The following paragraphs describe the design of the benchmarking modules for one or more forwarders when a consumer retrieves content from a producer across an nMANET.

System under test The initial step in the design of each experiment is to define the system under test and to isolate it from the environment that is not part of the system under test but may influence its observed behaviour.

System parameters The system parameters describe the configuration of the system under test, such as the forwarding strategy chosen, the size of the content store or the underlying transport to use.

Environment The environment and its parameters determine the way a forwarder is linked to its outside space, such as to other forwarders, proxies, consumers or producers. The parameters associated with the environment include factors such as the type of network communication with other nodes (wireless or wired networks), the bandwidth or error rate (in the case of a simulation).

Workload The workload in the benchmarking model is the aggregate of service requests to one nMANET node or a group of them. The workload can be divided into two main categories:

- **Synthetic:** A synthetic workload is created and designed to assess high coverage of a particular aspect of the system. The main synthetic workload used for nMANET experiments is represented by a file, for example, a text, PDF, or multimedia file with variable size. The parameters and format of the file can be changed during the experiments.
- **Real:** On the other hand, a real workload is generated from real working systems, such as common or popular applications of daily use.

Results The results represent the expected outcome of the experiment. This expected outcome is specified during the design and confirmed at the end of the data analysis, which is supported by the collected experimental data. In this chapter, the expected results include measurements of relevant evaluation metrics such as energy consumption and network traffic. The data analysis of results from experiments include evaluation of metrics such as *performance* and *cost*. In MANETs, performance measures correct functionality and time responsiveness of a system [Effelsberg et al., 2013]. They also measure the degree of quality of a task or operation executed in the system under test. In decentralised networks, such as MANETs, performance is expressed in terms of responsiveness, throughput and validity. This last one includes correctness and completeness.[Effelsberg et al., 2013], which are presented in the following paragraphs.

Correctness Correctness, in general ad-hoc communication, refers to whether the sent content was consumed as expected. Specifically, in an nMANET, it examines whether the

consumer receives the content correctly, as it was originally generated by the producer. The method used in this chapter to measure this metric is the ratio of the number of correct segments received by the consumer to the total number of segments sent by the producer, where correct segment means that the content of the segment in the consumer is identical to that sent by the producer.

Completeness Completeness, in ad-hoc communication, refers to whether sent content has arrived completely to the consumer. In an nMANET, completeness checks whether the number of segments sent by the producer is identical to that received by the consumer. This metric is measured as the ratio of the number of segments received in the consumer to the total number of segments sent by the producer. The comparison verifies that the name associated to the segments in the consumer is same as the ones sent by the producer.

In the context of a single nMANET node, cost refers to the utilisation of local resources when a node forwards NDN packets. Examples include energy consumed from batteries or networking resources consumed through the generation of network traffic. The cost for the network as a whole is a related but distinct concept. Sometimes, a larger cost for the wider network can be avoided by placing a higher cost on some individual nodes, e.g., by letting them use their storage capacity to cache content, which then does not need to be retrieved all the way from the producer, saving overall network resources.

As energy stored in mobile batteries is a limited resource, nMANET prioritises assessing forwarding strategies that spend less energy. Therefore, nMANET considers energy consumption as one important metric to measure the success of a forwarding strategy, which means that the less energy a forwarding strategy consumes, the more efficient it is.

6.2 Experiments overview

This chapter presents a series of experiments deployed on real devices and on simulated environments. At a high level, the experiments presented in this chapter can be divided into two main groups, one with a focus on network traffic and the other on energy consumption. The first group validates the nMANET framework using JNFD and HTTP. The reason why HTTP is chosen is that, like JNFD, it allows for the addressing of content, which a TCP channel alone does not. However, the HTTP cases have to use a routing mechanism for IP traffic to be able to reach distant nodes. OLSR is chosen for this as it has a widely available implementation, called olsrd2 [1]. The second group of experiments on energy consumption presents validation

[1] http://www.olsr.org/mediawiki/index.php/OLSR.org_Network_Framework

of the nMANET framework using JNFD, as well as other technologies such as OLSR and the traditional hotspot. AODV was not included because the online available implementations tested in Android mobile devices did not behave as expected, which was not the case for OLSR.

Table 6.1 summarises the experiments that belong to these two main groups. This table organises these experiments and provides a brief description of each them. The first group is a set of experiments where network traffic is measured in small and medium ad-hoc network topologies. To scale up the number of nodes in an experiment, Mini-JNFD is utilised to simulate a virtual MANET that uses GPS traces to provide mobility to nodes in the MANET. As an example the appearance of the user interface of the simulations, Figure 6.2 shows screenshots of the MANET at four different points in time, illustrating four changes to the topology over time. The consumer and producer nodes are highlighted to show how they move in relation to the other nodes.

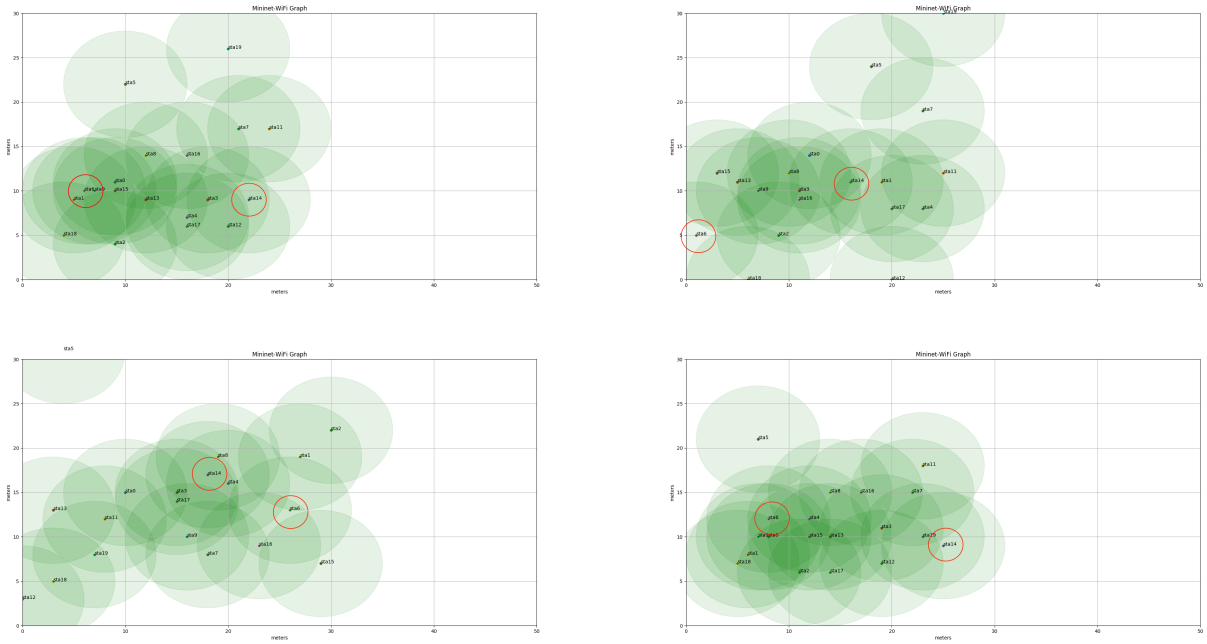


Figure 6.2: Four examples of visualisation of 20 mobile nodes of a simulated MANET using GPS traces.

The second group of experiments focuses on validation of the framework by measuring relative energy consumption from batteries. It starts by introducing the methodology of how energy consumption was measured in Android smartphones. It then validates the nMANET framework by using JNFD own forwarding strategies and OLSR routing features. It also utilises JNFD forwarding strategies and routing in infrastructure mode using a Wi-Fi hotspot. Finally, FIB next hop selection strategies are validated in order to estimate their relative energy consumption in devices such as Android smartphones.

Besides the general benchmarking model presented in Figure 6.1, the following sections and

Category	Experiment
Using simulations	Validation of data retrieval: request-response. - Data retrieval using different consumers. - Data retrieval using different forwarding strategies. - Identifying network traffic per node.
	Measuring network traffic of a MANET protocol. - Validation of data distribution across a network through caching.
	Evaluation of JNFD complementary features.
Using real mobile devices	Methodology for validation using real mobile devices.
	Baseline energy measurements.
	Validation of energy consumption measurements for data retrieval. - Energy consumption of JNFD and wireless Hotspot. - Energy consumption of JNFD and OLSR.
	Validation of data retrieval using JNFD forwarding strategies.

Table 6.1: Summary of main set of experiments

subsections describe each of these experiments, their expected behaviour, the methodology and infrastructure used to collect the experimental data and analysis of the results. The results from these experiments intend to show that, by using the nMANET framework, it is feasible to ensure responsible research on applying named data networking for content distribution in a MANET. Additionally, these experiments provide an evidence of reproducibility of the JNFD testing framework, as well as its limitations.

6.3 Validation using simulations

The experiments of this section validates the nMANET framework by using JNFD and HTTP in an IP-based network running the `olsrd2` routing daemon. In the following I will simply refer to HTTP where this protocol is used and omit explicit mention of OLSR. Experiments are run in Mininet-WiFi [Fontes et al., 2015] configured and controlled by Mini-JNFD (see Section 5.3). The rational behind these experiments is to measure the network traffic generated when consumers retrieve files from a producer (in the case of JNFD) or web server. All these experiments are scripted and therefore reproducible.

The experiments are executed in a virtual machine created by Mini-JNFD, which uses Vagrant to configure the virtual machines, their operating systems and installs Mininet-WiFi and `olsrd2`, an open source implementation of OLSR. Inside the virtual machine, the Testing and Experiment Manager (TEM, see Section 5.3) executes one of two Python scripts to set up the MANETs,

consisting of 20 nodes in Mininet-WiFi, corresponding to the only 20 usable GPS traces generated from the Crawdad data (see Section 5.1.1.3). The Wi-Fi network interfaces of all nodes are configured to run in ad-hoc mode.

The difference between the two scripts is that one configures nodes to run JNFD forwarders while the other configures `olsrd2` using the default configuration (as suggested on the OLSR website[2]). Once this is done, the scripts control the execution of the workloads in the MANET that was created as well as the execution of the GPS trace file that contains the sequence of Cartesian coordinates determining each node's movements over time.

In the case of JNFD, all nodes execute JNFD, producer nodes execute a JNFD producer, and consumer nodes JNFD consumer applications. For OLSR, the client nodes execute `curl`[3], while server nodes execute the Python `SimpleHTTPServer`[4] and all nodes run the `olsrd2` daemon.

The *independent variables* of these experiments are the size of the file on the producers/servers, the number of available files and the number of requests sent by the consumers/clients. For both cases, JNFD and HTTP, this workload is the same. The *dependent variables* utilised for JNFD and HTTP are the network traffic generated in the overall network as well as on individual nodes. In both cases the Unix `netstat` command is executed every second on each node throughout the experiment. The data generated is aggregated in the analysis to obtain network-wide measures.

6.3.1 Validation of data retrieval: request-response

6.3.1.1 Functional description

The purpose of this experiment is to validate the network traffic generated when consumers or clients retrieve files by using JNFD and HTTP in an IP-based network using OLSR. The expected result from this experiment is to provide evidence that name-based networking implementations, such as JNFD, can be used for content distribution in a MANET.

The information providers in these two cases make available 100 files of 10 Kbytes and another 100 files of 100 Kbytes size. The JNFD consumer and the HTTP client randomly select one of these files and send a request to retrieve its content. This is repeated 50 times for the smaller files and 30 times for the larger ones.

In both cases, mobile node number six (sta6) is selected as the JNFD consumer and in the case of HTTP as the web client. Mobile node 14 (sta14) is selected as the JNFD producer and, in

[2] http://www.olsr.org/mediawiki/index.php/OLSR.org_Network_Framework#olsrd2
[3] https://curl.haxx.se/
[4] https://docs.python.org/2/library/simplehttpserver.html

the HTTP case as a web server. These two nodes were selected due to their initial geographic location that allows them to have most of the rest of the nodes as intermediate forwarders. Their positions are highlighted in Figure 6.2.

Once the experiment starts, the `netstat -iec` command is executed on all nodes to collect network traffic measurements every second, which are analysed at the end of the experiment.

6.3.1.2 Data analysis

Once the experiment is finished, TEM retrieves the log files to the host machine for analysis. A sample of the output format is presented in Listing 6.1.

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 4  bytes 12406 (12.4 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 4  bytes 12406 (12.4 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

sta3-wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.0.0.4  netmask 255.0.0.0  broadcast 0.0.0.0
inet6 fe80::ff:fe00:300  prefixlen 64  scopeid 0x20<link>
ether 02:00:00:00:03:00  txqueuelen 1000  (Ethernet)
RX packets 437  bytes 241938 (241.9 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 399  bytes 408311 (408.3 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Listing 6.1: Sample output of netstat -iec collected per mobile node

A shell script filters out the data for the Wi-Fi interfaces, the IP address of the node and the number of bytes transmitted. This script also adds up the number of bytes transmitted for all nodes. This and the following paragraphs present the expected outcome as a result of the analysis of the data collected for this experiment. Figure 6.3 presents the results of plotting the overall number of bytes per second for JNFD using the 10k files (red) versus HTTP using the 10k files (purple) as well as JNFD using the 100k files (green) versus HTTP using the 100k files (light blue). The experiment with the larger files was run so as to ensure that the traffic generated exceeds the size of the Content Store on the nodes.

As expected, what is immediately striking about the diagram is that the overall network traffic generated for transferring the data is quite high. For transferring 30 times 100kBytes, or 3MB, the network nodes send 30MB worth of packets in the HTTP case. This is due to a number of

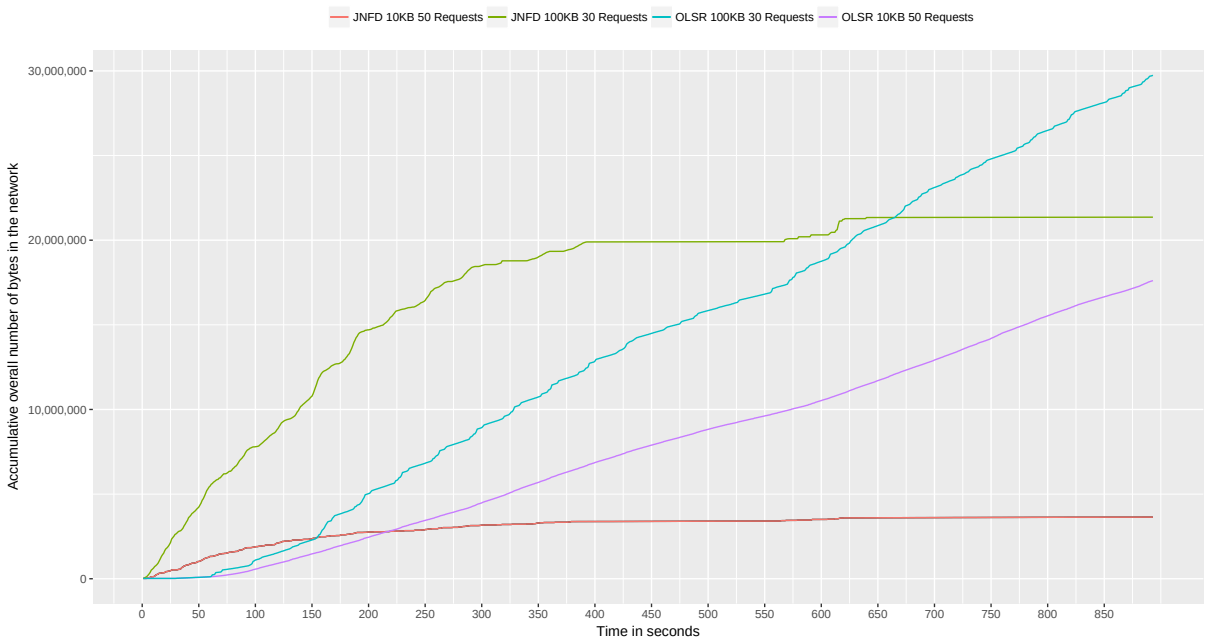


Figure 6.3: Overall aggregated network traffic JNFD vs HTTP

factors: first, as data communication in a MANET is hop-by-hop and I am measuring the total size of packets sent by each node, the aggregate is influenced by how many hops a data packet needs to make to reach its destination. The overall traffic for the JNFD case is lower but still many times the size of the workload.

Considering the case of JNFD, I analyse Figure 6.3 in two parts: the first one points to the area where the traffic is higher in JNFD than HTTP, and the second part where the traffic of JNFD has a flat tendency while the HTTP traffic keeps rising. The first part is generated due to the traffic generated by intermediate forwarders that are searching for the content requested by the consumer. This traffic is high at the beginning due to the forwarders broadcasting the interest until they reach the producer, and due to the data packets being sent to the consumer using the reverse path. However, as JNFD caches the content, the next request of already cached content is served by the forwarders and not necessarily by the producer. Furthermore, as the effective path to content is shorter and as nodes have information in the FIBs, the number of interest broadcasts will be reduced. Consequently, the network traffic after the content is cached in intermediate forwarders is reflected in the curve flattening out.

Additionally, and from the consumer point of view, this experiment shows that the time required to deliver the content varies according to where it is coming from. On the one hand, it corresponds to the time utilised when the content comes from the producer. On the other hand, it corresponds to the time utilised when the content comes from the cache of nearby forwarders. For example,

the time required to retrieve a 100KB file from the producer is in average 2773.167 milliseconds, while in the case that the same file is retrieved from the cache of forwarders, this time is reduced to 7.975 milliseconds, in average. This last case highlights the benefits of provide caching to forwarders in order to reduce latency.

Another benefit of the use of caching in JNFD is reflected in the level of completion rate of the requested content. As a example, retrieving 50 times a randomly selected file of 100KB size from a producer, has a completion rate of 71%, in JNFD, while in OLSR this value is 67%. This supports the idea that content distributed across forwarders offers more availability of the information, and less dependency of the producer.

In the case of OLSR, the network traffic rises linearly. For both file sizes, the network traffic is constantly increasing. A degree of flattening of the curve could be expected if OLSR was benefiting from using established routes after a period of operation, dealing only with incremental changes to the topology. However, this does not seem to be the case, which raises the question whether the overhead of OLSR routing traffic decreases over time if there is no other traffic and if the nodes in the MANET are static. The following paragraphs present the result of OLSR network traffic generated in a MANET where nodes are following GPS traces, which is the current behaviour, and where nodes are standing at fixed locations and where no workload traffic is being generated.

6.3.1.3 Overall network traffic validation

Figure 6.4 shows the results of OLSR isolation and network traffic measurements in two scenarios. The first one corresponds to the OLSR network traffic when `olsrd2` is executing in nodes that have mobility through GPS traces files. The second case is the network traffic when `olsrd2` is executing in nodes that are static.

In both scenarios only the `olsrd2` daemon and the `neststat` commands are executed on each node. Consequently, only OLSR traffic can be observed and measured. Figure 6.4 presents the results of these measurements in both scenarios. As it is expected and from the graphs, the overall traffic is constantly increasing over time, even when the nodes are not moving from a fixed position. This suggests that the overheads of using OLSR do not diminish much if the nodes stand still. This is an unfortunate consequence of OLSR's proactive approach and its inability to identify cases where nodes are static and the level of routing traffic could perhaps be reduced.

As it was expected, this complementary experiment confirms that OLSR network traffic constantly generates traffic and so contributes significantly to the network traffic shown in

Figure 6.3. The experiment with OLSR running without a workload shows that it generates about 4MB of traffic on its own over the course of the experiment, which is about 7,207 kBytes per second for the whole network or 360 Bytes per second per node.

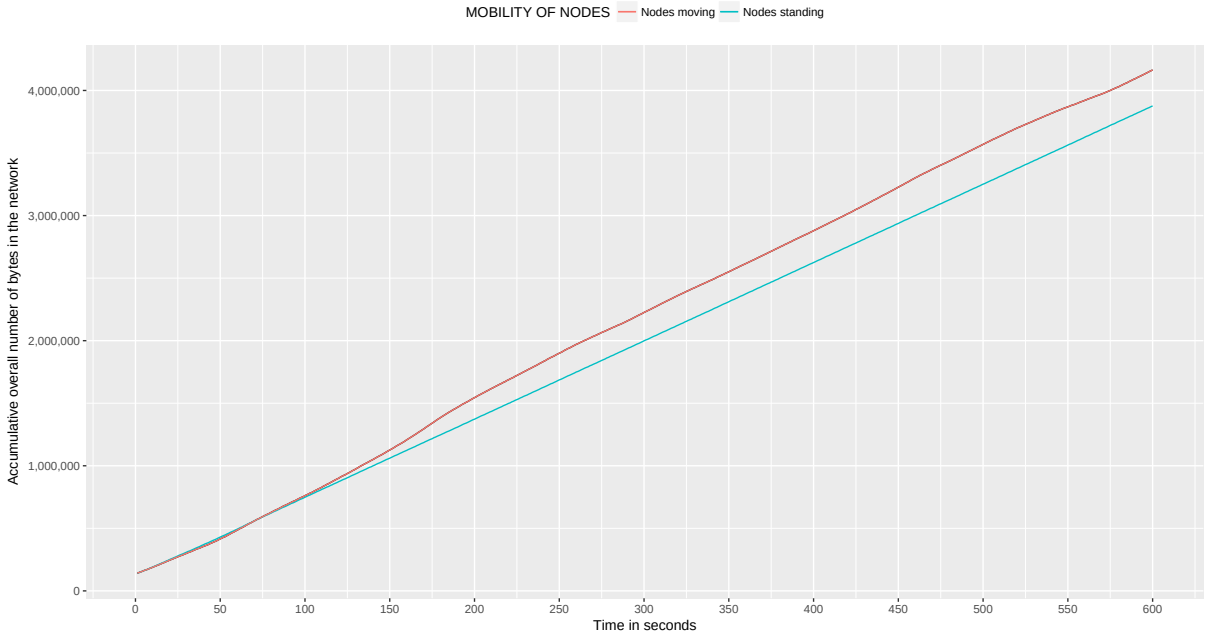


Figure 6.4: Comparing overall OLSR network traffic: nodes standing vs nodes moving with GPS traces

OLSR convergence time During the experiments with HTTP, the scripts that execute the simulations with OSLR include a waiting time before triggering the request from the web client. This is visible in Figure 6.3 at the start of the blue line, and was introduced because OLSR requires a time for convergence of the routing tables in the nodes of the MANET. On the other side, JNFD does not require this waiting time as JNFD does not depend on being able to construct an end-to-end path between the data source and destination.

6.3.1.4 Data retrieval using different consumers

From the JNFD perspective, this experiment supports that the network traffic generated by different nodes chosen as the consumers behaves similarly, which is an expected result. Figure 6.5 presents the results of the overall network traffic generated when one of six different consumers requests a random file 20 times. As expected, the results support that JNFD produces the same behaviour, high network traffic at the beginning with a flat behaviour after data is cached. Note that each curve on Figure 6.5 is associate with a respective consumer and a completion rate. The differences between these curves is due to each consumer starts in different locations and each consumer has different GPS traces. However, all of them have similar behaviour.

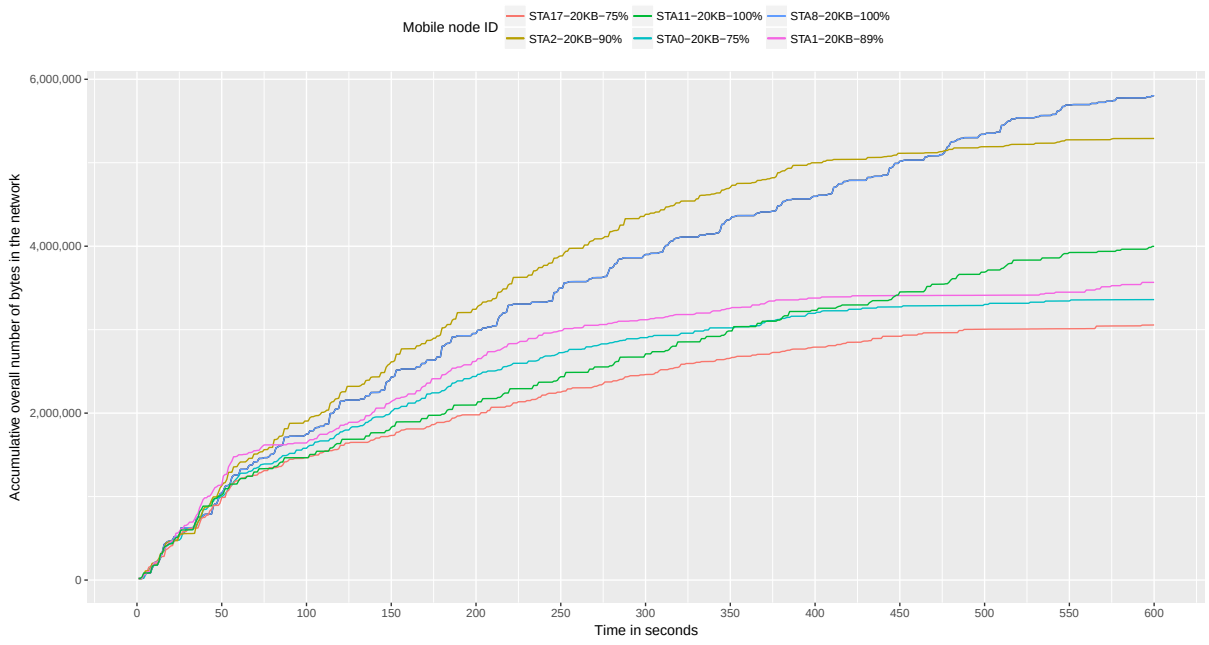


Figure 6.5: Cumulative overall network traffic of different consumers requesting to the same producers using JNFD

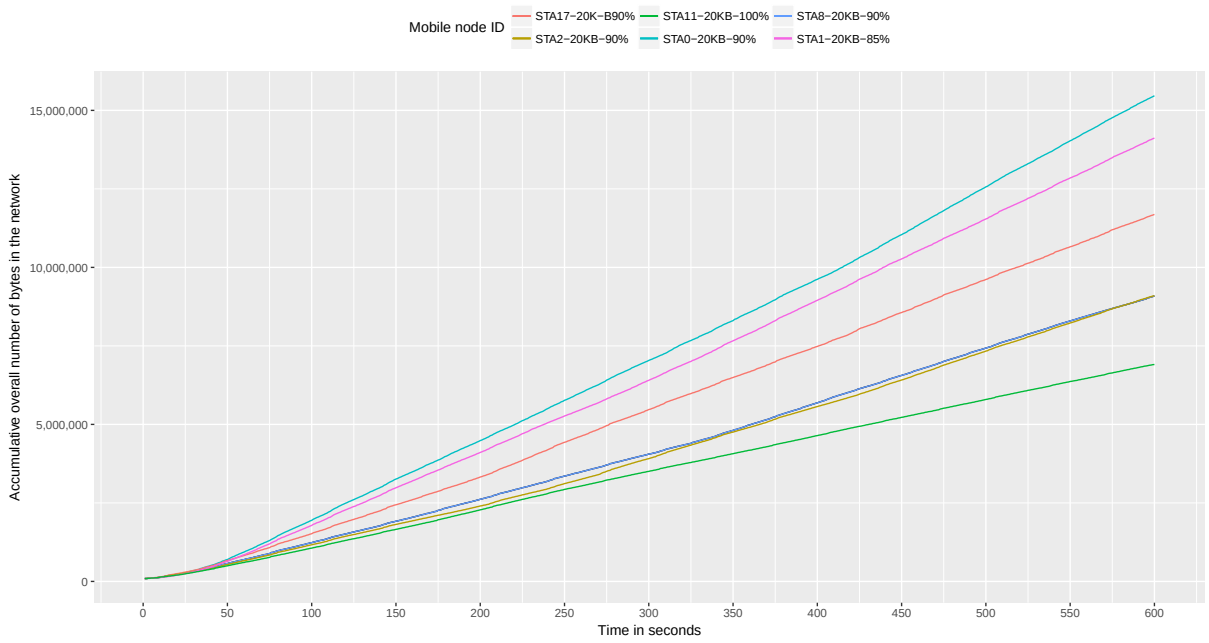


Figure 6.6: Cumulative overall network traffic of different clients requesting to the same web server using OLSR

6.3.2 Data retrieval using different forwarding strategies

In previous experiments, forwarders execute the default FIB next hop selection strategy, which is unicast. In this complementary experiment, JNFD forwarders are set to use the random and broadcast strategy in order to compare the network traffic of these two forwarding strategies against unicast. At the time of this writing, setting additional arguments when a node is created in Mininet-WiFi is still not available. For example, it is not possible yet to assign initial battery level and depletion model to mobile nodes. Therefore, this complementary experiment does not include forwarding strategies such as battery, storage or geo-location. As this is a limitation of Mininet-WiFi, this chapter includes energy consumption measurements on real mobile devices in Section 6.4

Figure 6.7 shows high network traffic generated by the broadcast forwarding strategy in comparison to unicast and random strategies. However, the percentage of completeness of the random strategy is lower than the broadcast. This experiment not only presents the expected results from comparisons, it also represents a reference and exemplifies the methodology that JNFD suggests to use at the time to test new forwarding strategies. Section 5.7 provides guidance on how new forwarding strategies can be integrated into JNFD and this experiment provides one methodology to validate existing JNFD forwarding strategies.

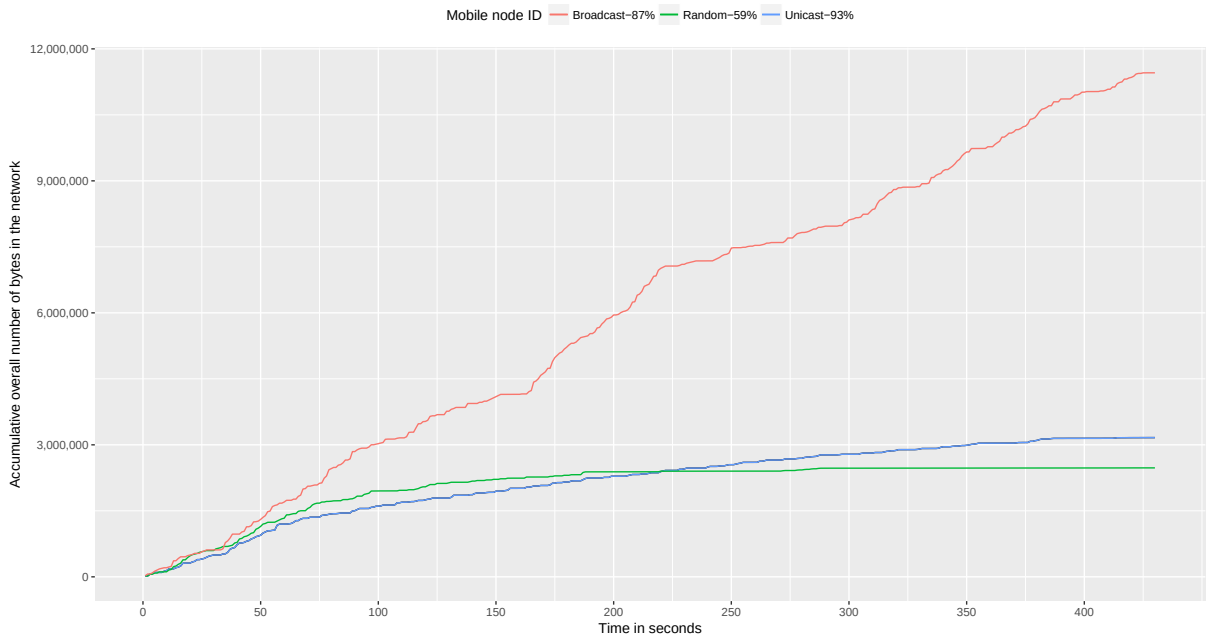


Figure 6.7: Cumulative overall network traffic of JNFD MANET for random, unicast and broadcast strategies. Label includes percentage of completion.

6.3.3 Identifying network traffic per node

Previous results validated that the network traffic generated by JNFD is less than the generated by OLSR. As this is an overall result, the necessary next step is to analyse the network traffic behaviour one layer deeper, at the mobile node level. This complementary experiment identifies the overall traffic generated by each node on JNFD MANET and on HTTP MANET.

The `netstat` command records the cumulative traffic per interface every second. I wrote a script that filters and extracts the cumulative network traffic at a similar point in the time for all nodes and when the experiment is just about to finish. The procedure consists of extracting the last records of the netstat log files and ordering them by the number of transmitted bytes through the wlan0 interface of each node. Figure 6.8 presents the results of the data processing of the netstat logs for each node. This table presents the cumulative network traffic per node at the end of the experiment.

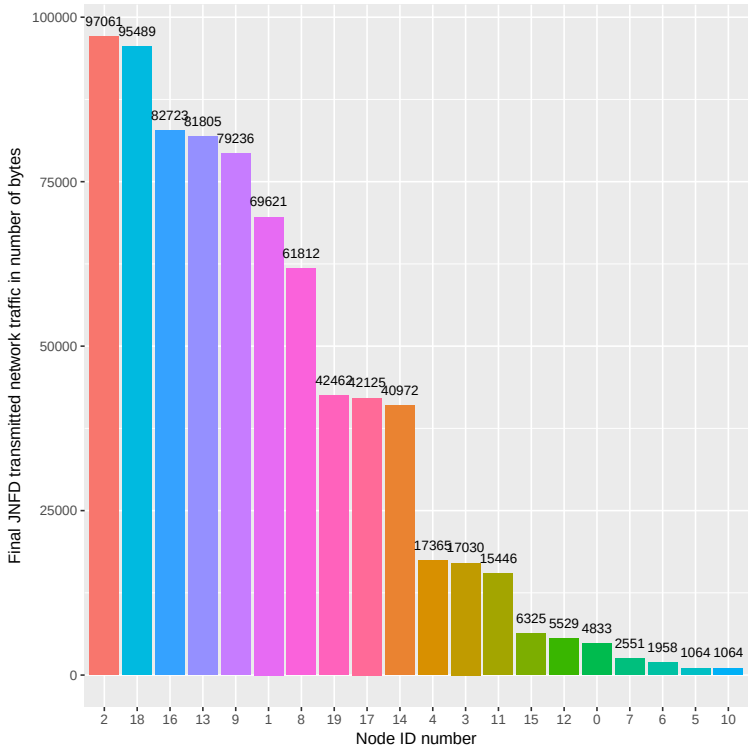
Note that the JNFD producer sta14 (with log file `netstat14.jnfd`) has less transmitted traffic than its respective web server node in OLSR, the web server generated a traffic at least four times higher than the producer in JNFD MANET. The mobile node sta5 is an isolated node that most of the time is disconnected from the rest of the nodes. Therefore, it is to be expected that this node generates minimal traffic. Even, it is the one that produces less traffic, there is a considerable difference between the network traffic that this node produces in HTTP MANET compared with its network traffic in JNFD MANET. This can be confirmed by the received traffic per node presented in Figure 6.9.

The ranking of transmitted and received network traffic presented in Figure 6.8 and 6.9 identify the nodes that do not participated in the retrieval of the content, such as the case of node 5 and 10. Additionally, these two figures show that both incoming and outgoing traffic are distributed differently. In JNFD MANET there are about 13 nodes that generate and received more traffic, ranking position 1 to 13, while in HTTP MANET there are 9 nodes that generate and receive more traffic, ranking position 1 to 9. This behaviour gives the idea that the data is distributed more uniformly in nodes of the JNFD MANET than HTTP MANET. Consequently, this validates that content also has more uniform distribution in JNFD MANET.

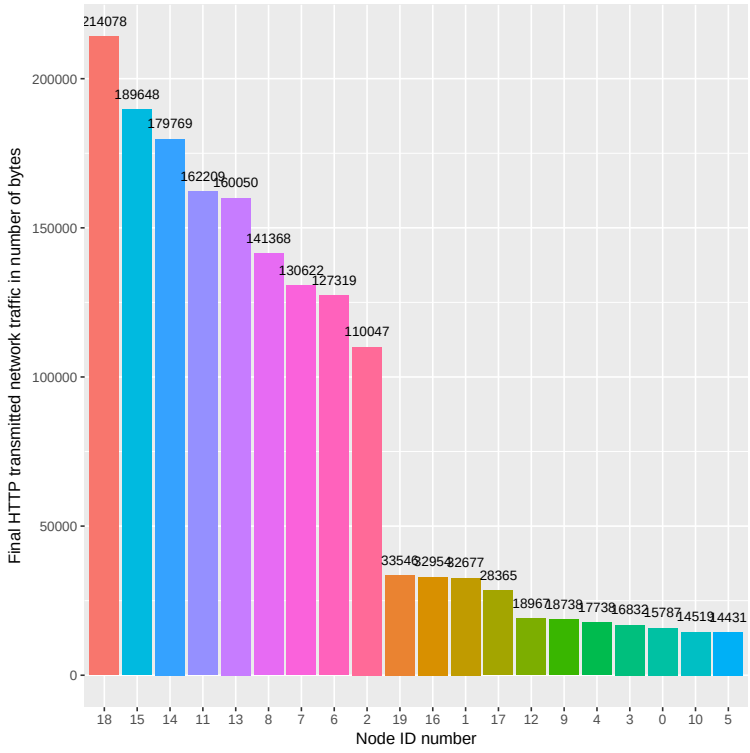
6.3.4 Measuring network traffic of a MANET protocol

6.3.4.1 Functional description

This experiment validates the network traffic generated by two MANETs, one that uses JNFD, JNFD MANET, and another that use OLSR only, OLSR MANET. In both cases, the

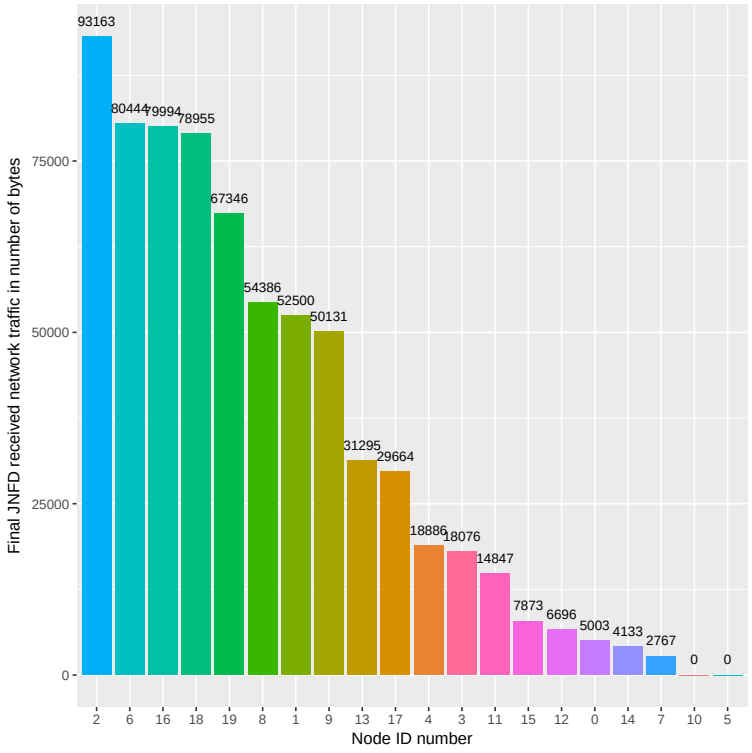


((**a**)) Ranking based on overall JNFD transmitted network traffic

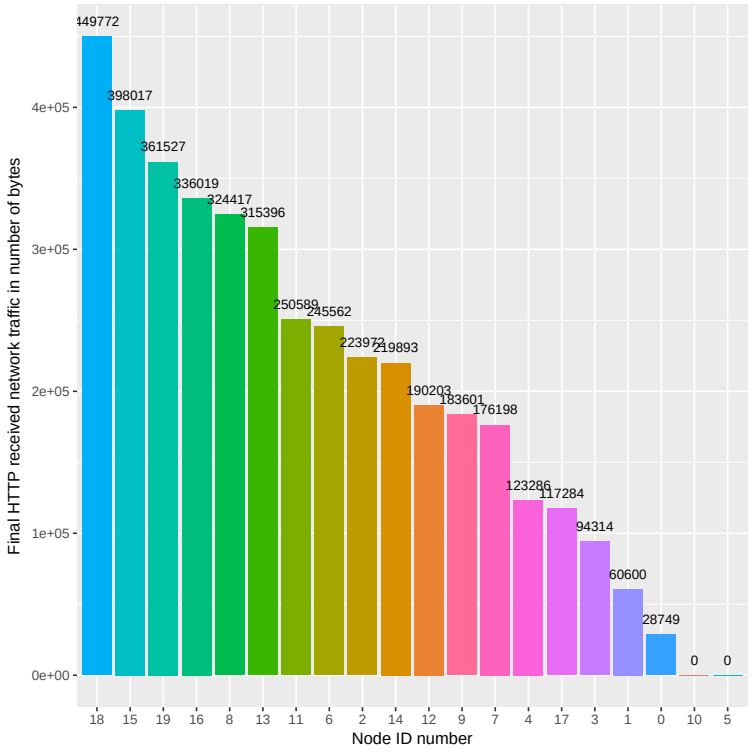


((**b**)) Ranking based on overall HTTP transmitted network traffic

Figure 6.8: Ranking of number of bytes transmitted per node at the end of the experiment



((a)) Ranking based on overall JNFD received network traffic



((b)) Ranking based on overall HTTP received network traffic

Figure 6.9: Ranking of number of bytes received per node at the end of the experiment

consumer/client requests a file from a remote producer/server. This experiment observes the behaviour of the generated network traffic of JNFD MANET and OLSR MANET. In this experiment the OLSR client utilises the same application as the JNFD consumer, and similarly, the OLSR server executes the same application as the JNFD producer. The rest of the nodes in JNFD MANET are JNFD forwarders and in the case of OLSR are olsrd routers. Having the same application in both networks, and the same type of transportation, the expected results from this experiment validate that OLSR, carrying NDN messages in JNFD, generates less traffic than HTTP MANET.

In other words, the difference with previous experiment, Section 6.3.1, is that in the case of the OLSR MANET, the client node number six is not a web client, as it was in HTTP MANET, and also the mobile node 14 is not a web server. Instead, the client node six executes a JNFD consumer, and the server, node 14, executes a JNFD producer. The consumer makes 10 requests of a file that is randomly selected from 100 files of 10Kbytes size located in the producer. Note that in the case of OLSR MANET, the JNFD consumer requires to knowing the IP address of the JNFD producer in order to retrieve the content, which is not the case for JNFD MANET.

6.3.4.2 Data collection and analysis

The results of this experiment are shown in Figure 6.10. This figure plots the overall network traffic in both cases. As it was expected, the curve of the cumulative network traffic of JNFD has a similar shape as in previous experiments, high traffic at the beginning that have a flat tendency along the time.

Analysing of OLSR web client server As from Figure 6.10, the overall network traffic of OLSR MANET has similar behaviour as previous experiments, Figure 6.11 presents an insight to one of the immediate doubts, which is to verify whether the network traffic of OSLR MANET is similar to HTTP MANET. For this purpose, I make a copy of the script from OLSR MANET and set the client to use `curl` as a web client, instead of the JNFD consumer, and the server to a web server, instead of a JNFD producer. In this way, it is possible to observe the behaviour of both networks, HTTP MANET and OLSR MANET. The results are presented in Figure 6.11, and it appears that both cases have similar behaviour, a constant increase in the network traffic and an apparent linear tendency along the time. This figure also shows that OLSR MANET, using JNFD consumer/producer, generates about half less of the transmitted network traffic of a web client/server HTTP MANET.

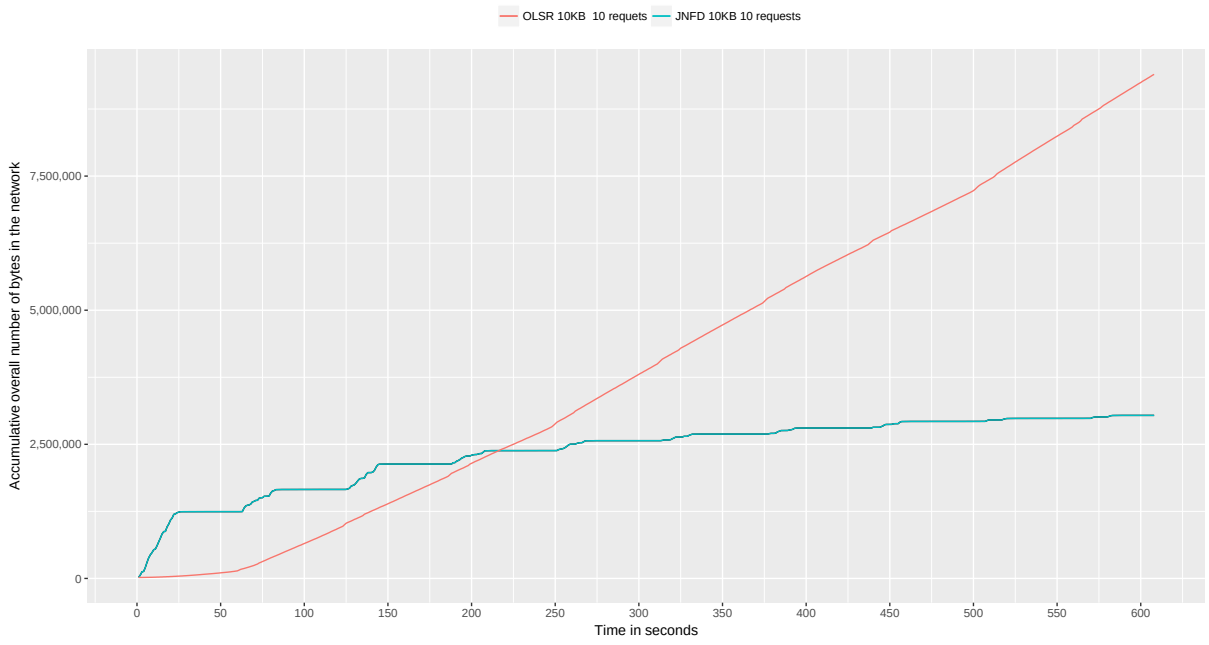


Figure 6.10: Network traffic in number of bytes: JNFD MANET and OLSR MANET. File to retrieve 10KB, Number of request =10

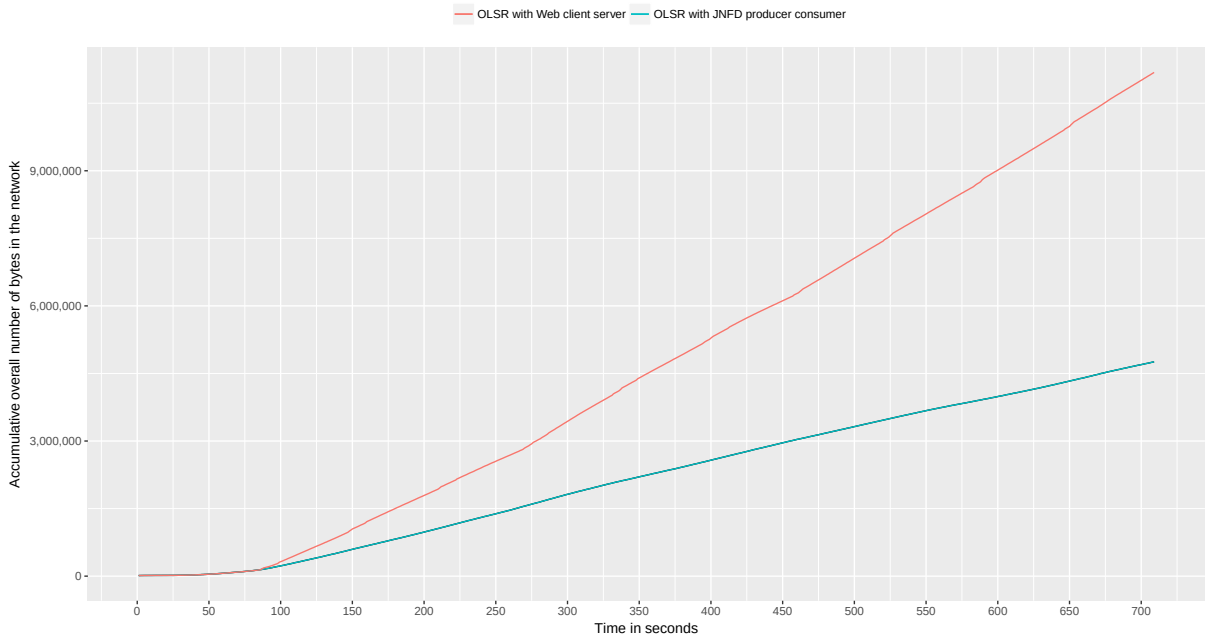


Figure 6.11: Network traffic of OLSR MANET using a JNFD producer consumer versus OLSR MANET using web client server

6.3.4.3 Validation of data distribution across a network through caching

This complementary experiment intends also to validate a methodology of analysis of how content is distributed in nodes in the case of the JNFD MANET, which refers to the use of cache in the forwarders. For this purpose, the consumer is required to send interest packets with the same name several times, as then the forwarders can reply the content by reading from their cache. In this complementary experiment, the consumer sends interests which name is selected randomly between 5 name files, instead of the 100 files as it was initially.

```
Attempt_Number      Name of the file  requested by consumer6
------------------------------------------------------------
10,8,9        jnfd:/a/b/files/files10K/file10K.1.txt
5,6           jnfd:/a/b/files/files10K/file10K.2.txt
2,4           jnfd:/a/b/files/files10K/file10K.3.txt
1             jnfd:/a/b/files/files10K/file10K.4.txt
3,7           jnfd:/a/b/files/files10K/file10K.5.txt
```

Listing 6.2: List of files randonly selected by the consumer number 6

List 6.2 presents the number of times that a consumer request interests with the same name. For example, the name "jnfd:/a/b/files/files10K/file10K.1.txt" was requested three times by the consumer in the attempt number 8,9,10. All of these requests were satisfied with 100% completeness and correctness. Figure 6.12 presents the cumulative overall network traffic of JNFD MANET during this experiment.

The number of times that a forwarder has received a data packet and send it back to the node that previously request it can give an idea of how the content is distributed in the cache of forwarders of JNFD MANET. Figure 6.13(a) shows the number of times that each forwarder use the reverse path to redirect a data packet, which can be understood as the number of times that data packets were also cached in the CS. This is due to an nMANET forwarder makes locally a copy of the received data packet in CS, Figure 4.4.

Figure 6.13(a) provides an idea of the distribution of the number of elements stored in the CS of forwarders. However, it does not shows the number of times that the CS was used to satisfy interests which names match the names stored in the CS. To have an idea of the use of the CS, I wrote a script that searches on the logs that forwarders record during the experiment to totalise the number of times that the interest was satisfied by the local cache. Figure 6.13(b) presents the number of times that a forwarder used its CS to satisfy an interest which name was already stored in its CS. For example, the forwarder with id 16 and 13 are the two nodes that used their CS more often.

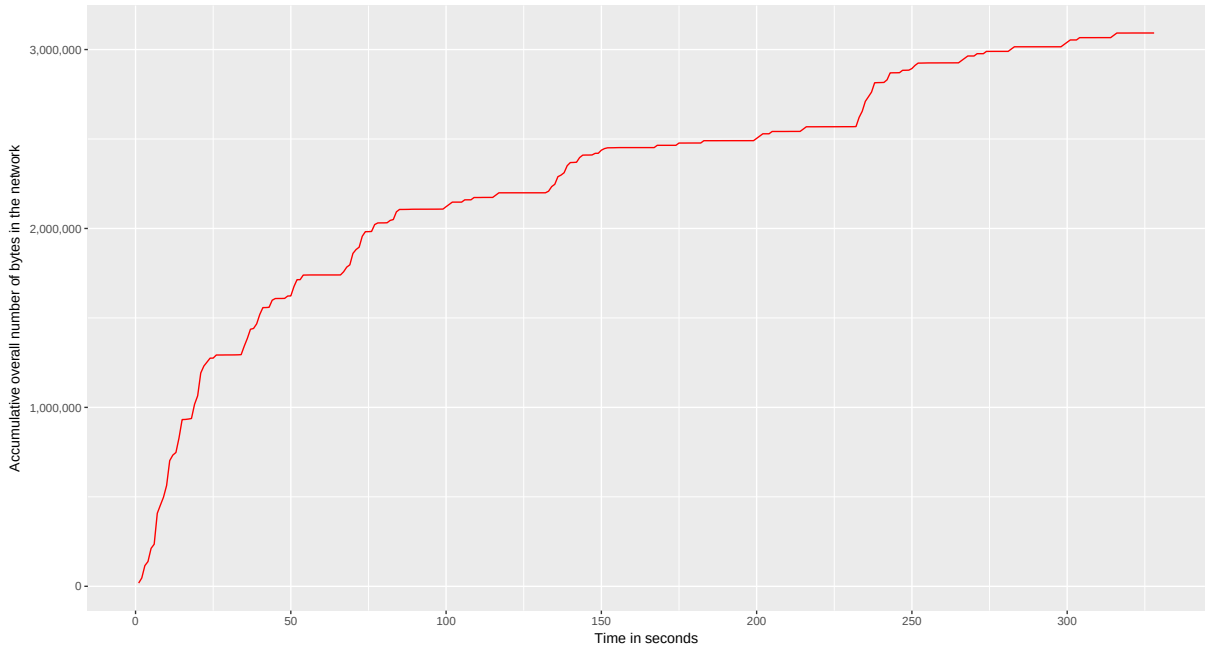


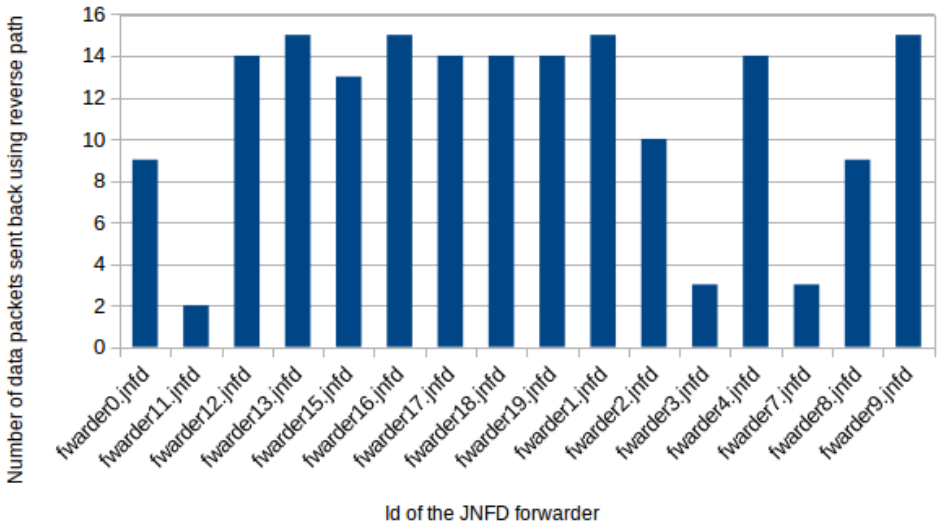
Figure 6.12: Network traffic of JNFD MANET for a consumer selecting randomly between 5 files

By combining this results with the network traffic ranking per node presented in Figure 6.14, is possible to identify critical forwarders that required more attention and resources. This is important in emergencies scenarios because these nodes carry most of the traffic in the network and are necessary to maintain the data distribution to new requesters. For example, it is expected that node 16 and 13 are the forwarders with higher network traffic during the experiment, and these two forwarders are also the ones that report higher access to CS to satisfy interest requests, as per Figure 6.13(b). The combination of these results confirm that these two forwarders have relevant impact on satisfy interest requests without retrieving the content from the producer.

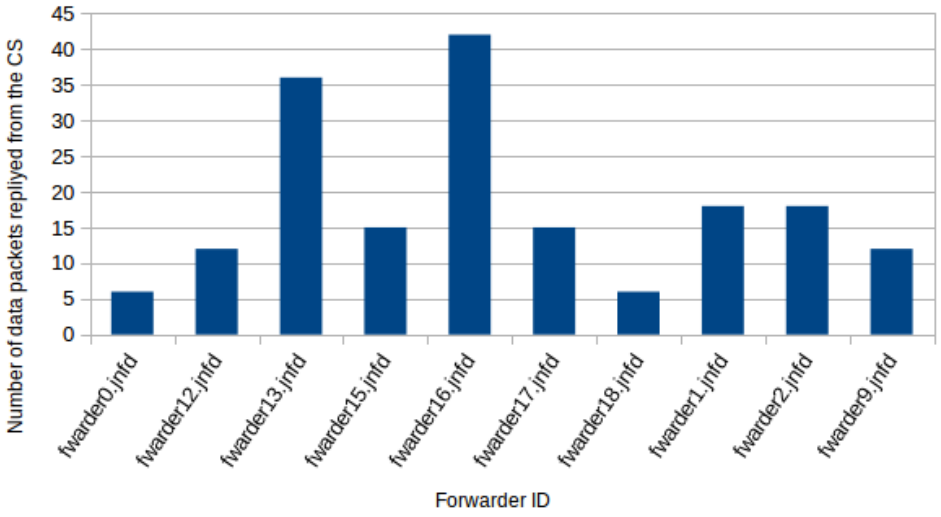
6.3.5 Evaluation of JNFD complementary features

6.3.5.1 Functional description

The purpose of this experiment is to validate the behaviour of the overall network traffic of a JNFD MANET when one of the complementary features of JNFD is enabled. This subsection presents the behaviour of the prefix propagation feature in JNFD forwarders, which is described in more detail in Section 4.7.4. However, other current or future features can be validated similarly. The aim of this experiment is to validate the network traffic behaviour when a JNFD forwarder has the prefix propagation feature enabled. The assumption at this point is that by initially propagating the registered prefix, the forwarders have an up-to-date FIB table that can satisfy future interests of names that matched the propagated prefix with less network traffic.



((a)) Number of data packets sent back by a forwarder using the reverse path



((b)) Number of data packets replied from the CS of forwarders

Figure 6.13: Number of data packets sent and received per node

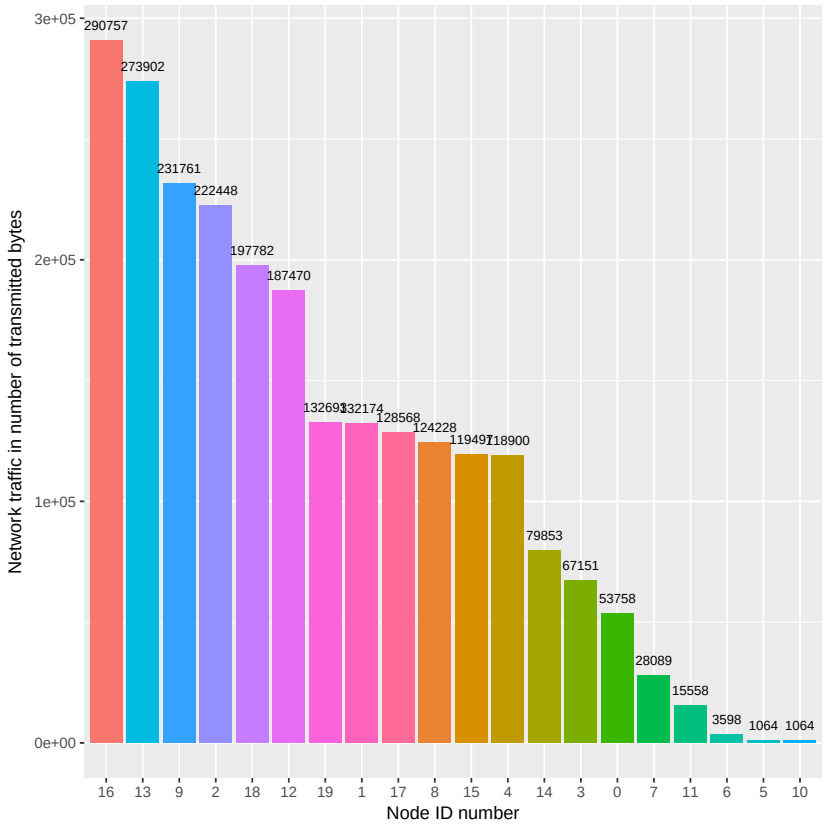


Figure 6.14: Ranking of network traffic per node in number of transmitted bytes

Consequently, the forwarders can execute the prefix discovery fewer times than in the case of not using prefix propagation, which means less broadcasting of interest packets.

This experiment presents the results of the execution of two scripts. One that builds a JNFD MANET where forwarders make use of the prefix propagation option and the other script when they do not use it. In both scripts, nodes 4 and 16 act as consumer and producer, respectively, and they were selected due to their initial geographic location that allows them to have most of the rest of the nodes as intermediate forwarders. The producer can serve 100 files of 10Kb each, and the consumer selects randomly one of the names of the 100 available files and sends interest requests to retrieve their content.

Finally, in both scripts, the `netstat -iec` command is executed in order to collect network traffic measurements, which are stored in log files and analysed in the next paragraphs.

6.3.5.2 Data collection and analysis

Figure 6.15 presents the results of the data analysis of the logs generated by the scripts of both experiments. This figure shows that the use of this feature generates more than twice the

traffic generated if prefix propagation is not used. The behaviour of network traffic generated by utilising the prefix propagation seem similar to the generated by the broadcasting strategy of the experiment in Section 6.3.1.2. Both cases suggest that broadcasting interest from the consumer or prefixes from the producer is not the best idea to minimise the network traffic in JNFD MANETs.

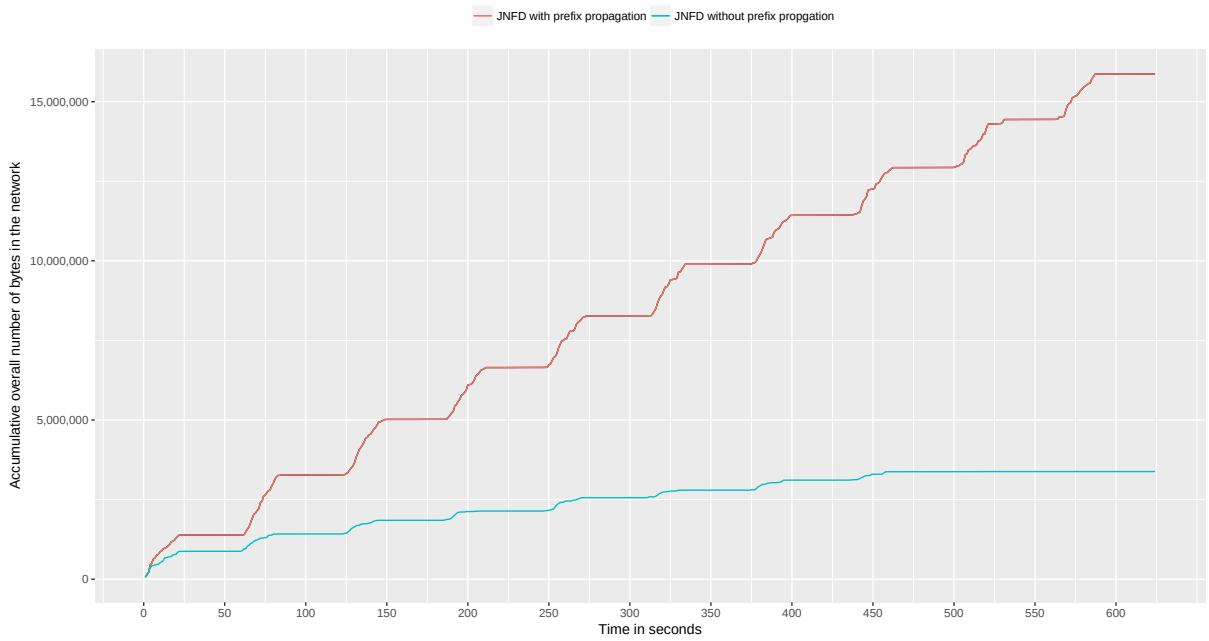
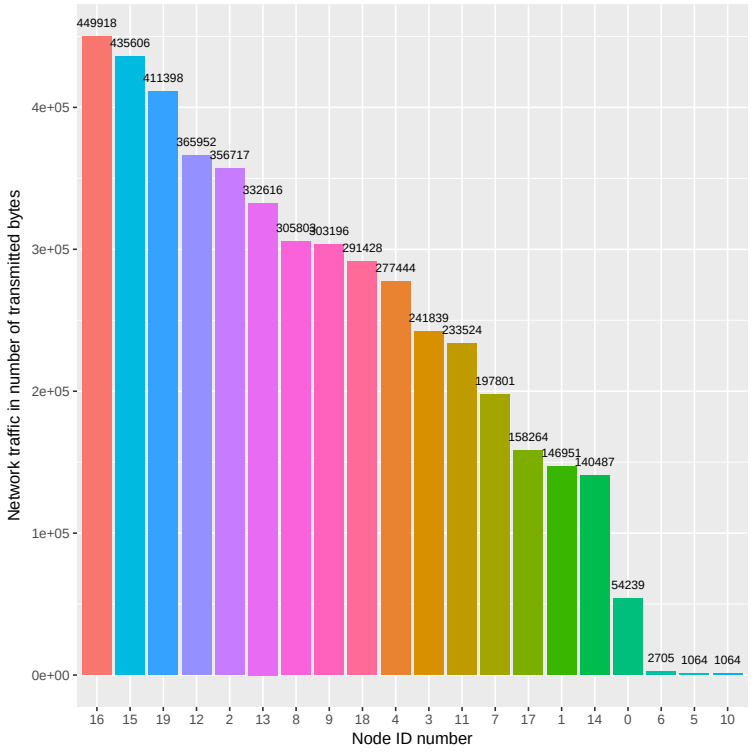


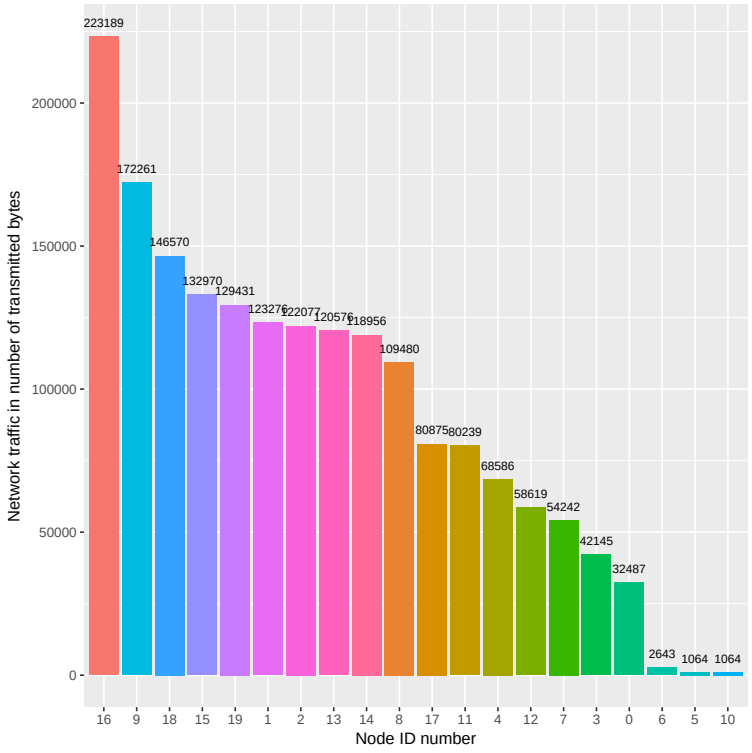
Figure 6.15: Network traffic of JNFD MANET with and with out prefix propagation, completion rate 100%.

However, there is one apparent advantage of prefix propagation in benefit of the producer. Figure 6.16 shows that the producer, in a JNFD MANET that uses prefix propagation, is one of the nodes that generates less traffic compared with another 15 forwarders, Figure 6.17. On the other hand, a producer, where forwarders do not enable prefix propagation, belongs to a group of nodes that generates more traffic. This can be understood as that the use of prefix propagation reduces the outgoing traffic on the producer. However, as it is expected and show in the following, it increases the overall traffic in the network.

Both graphs in Figure 6.16 show that the most five active nodes are 16, 15, 19, 12, and 2, which raise the question if their traffic is due to prefix propagation. As in JNFD, prefix propagation is implemented by broadcasting node status, which carries the list of available prefixes in the FIB of forwarders, I wrote a script that searches and counts the number of incoming node status packets to a forwarder. The List 6.3 shows a ranking of the nodes by the number of data packets received. From this list, the nodes 16, 15, 19, 12, and 2 are also in the top five of this ranking.

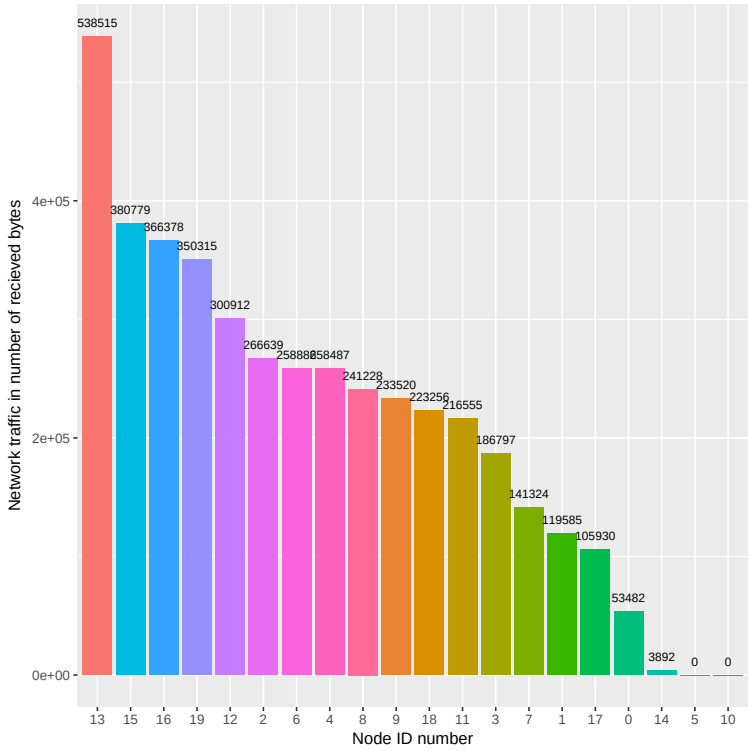


((a)) Ranking of nodes based on the transmitted network traffic with prefix propagation.

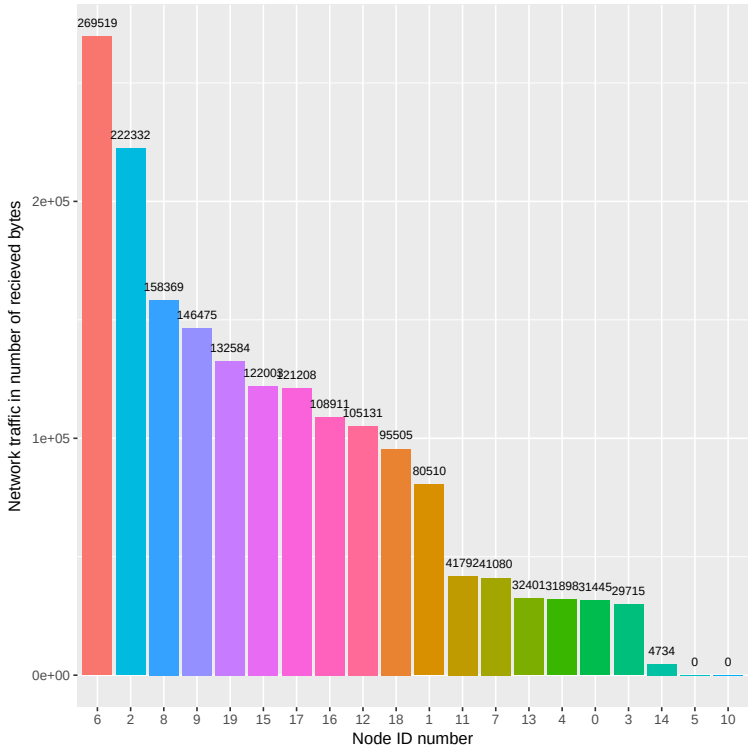


((b)) Ranking of nodes based on the transmitted network traffic without prefix propagation

Figure 6.16: Number of data packets sent per node with/without prefix propagation



((a)) Ranking of nodes based on the received network traffic with prefix propagation.



((b)) Ranking of nodes based on the received network traffic without prefix propagation

Figure 6.17: Number of data packets received per node with/without prefix propagation

```
1 fwarder15.jnfd  111
2 fwarder19.jnfd  105
3 fwarder16.jnfd  103
4 fwarder2.jnfd 97
5 fwarder12.jnfd  97
6 fwarder3.jnfd 96
7 fwarder4.jnfd 89
8 fwarder8.jnfd 88
9 fwarder13.jnfd  82
10  fwarder9.jnfd 75
11  fwarder18.jnfd  73
12  fwarder7.jnfd 58
13  fwarder1.jnfd 55
14  fwarder11.jnfd  54
15  fwarder17.jnfd  51
16  sfwarder0.jnfd  40
```

Listing 6.3: Ranking of nodes by the number of received node status packets

This combination of results give the idea that the traffic generated by prefix propagation is due to the broadcasting of the data packets between forwarders.

This raises the question of what leads to the increase in data packets sent in the prefix propagation case. One explanation would be that prefix propagation finds more feasible routes than prefix discovery.

6.4 Methodology for validation using real mobile devices

The following paragraphs discuss the approach taken to measure energy consumed by mobile devices. Quantifying the energy consumption of an application is not a trivial problem, and may require sophisticated equipment, such as digital oscilloscopes [Friginal et al., 2011]. This thesis utilises a relatively simple technique to measure energy usage in mobile devices during experiments.

6.4.0.3 Methodology for energy consumption measurements

The most basic technique to measure the flow of energy charges utilises a basic serial circuit, direct method, where an ampere meter (Ameter) or multimeter is connected in serial with a load of resistance R and a voltage supplier V. Using this configuration, depicted in Figure 6.18, the Ameter is able to measure the amount of electric charge in Amperes flowing through the closed circuit. This figure also highlights the analogy of this configuration using a mobile device with one condition, which is that the ammeter should be between the battery and the rest of the mobile device.

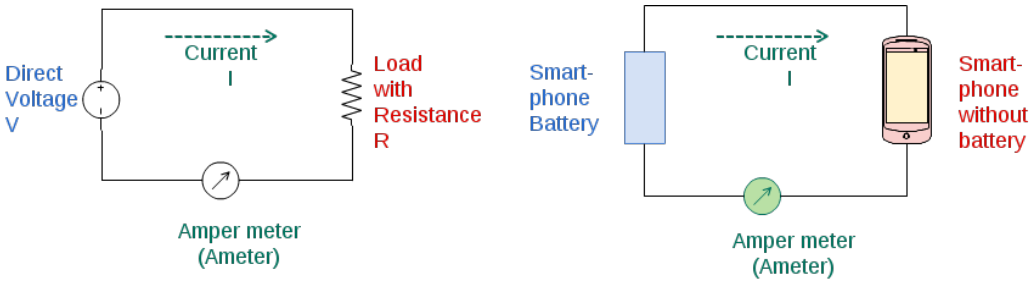


Figure 6.18: Direct method to measure energy consumption on a removable battery of a smartphone

Direct measurements are suitable in cases when the battery can easily be removed from the mobile device. In cases when the battery is locked firmly under a shell, such as the case of the smartphones used in experiments of Section 6.5, the Motorola MotoE XT1021 series[5], the covers and shells have to be removed every time a measurement is taken. For example, Figure 6.19 shows that the battery in MotoE XT1021 cannot be easily removed as a number of T4 Torx screws need to be removed and the wire between the battery and the printed cirquit board needs to be cut to insert the Ampere meter. Figure 6.20 shows the main infrastructure used to collect energy data through the direct technique.

Figure 6.19: Motorola MotoE XT1021 battery exposed

The yellow circles tagged with letter C and D in Figure 6.20 are the two cable extensions that allow connecting serially the multimeter with the battery and the main board of the MotoE as it is described in figure 6.18. Therefore, the multimeter can measure the amount of energy charges flowing from the battery to the smartphone's main board. The measurements and units displayed by the multimeter are tagged with letter A and B, these values are sent to a laptop through the

[5]https://sites.google.com/site/meshawaremobilenetworking/infrastructure/motorola-motoe-xt1021-specs

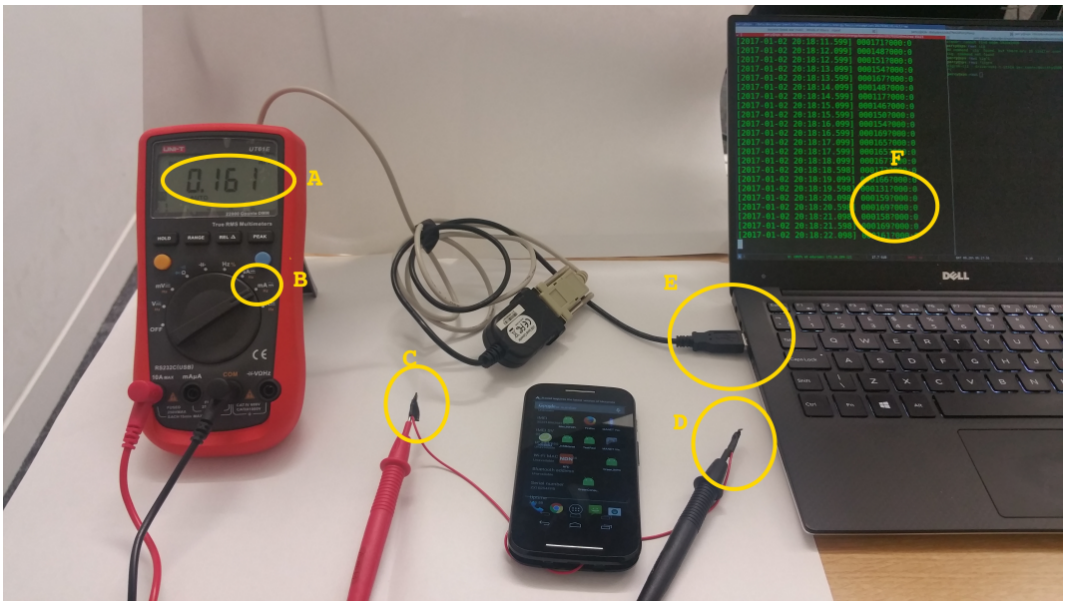


Figure 6.20: Infrastructure used to measure energy consumption directly.

infrared-RS232-USB cable (circle E). The laptop stores the received measurements in its internal file system (circle F) using the minicom application[6].

The following subsection presents an example of how this method was applied to measure the energy consumption of a camera application in a MotoE smartphone. This method is used in the experiments of energy consumption measurements presented in the following paragraphs.

6.4.1 Baseline energy measurements

The purpose of this subsection is to introduce the methodology to measure energy consumption in mobile devices, such as is the case of the Android smartphones utilised in the three experiments of Section 6.5.

This experiment estimates the energy consumption of a camera application by comparing the energy consumption baseline of a mobile device and the energy consumption when the camera is recording a video. Camera application was selected due to it is a popular application provided by most of mobile devices and easy to deploy.

The energy consumption baseline is estimated by setting a mobile device in its minimal conditions, which is after boot has finished and before the user interacts with the screen to run applications. This baseline is compared against the measurements of the energy when an application is being executed. Therefore, it is possible to estimate the statistical significance of the difference between

[6]https://help.ubuntu.com/community/Minicom

two data sets. This methodology is utilised in further experiments to compare two or more cases and estimate the statistical significance between them.

The following experiment collects data in two main stages, ”initial conditions” and when the user ”turns the camera on”. In this experiment, both stages are compared statistically to estimate the energy consumed by the application Motorola camera (“com.motorola.camera”). The two stages to compare are summarised as follows.

1. **Initial conditions** In this stage the mobile is set to its initial conditions. After turning on the smartphone, the GPS and Wi-Fi are turned off, and the Firefox application process killed manually.

2. **Turning camera on** Starting from the ”initial conditions”, the user then turns on the Motorola camera application only, and starts to record a video.

6.4.1.1 Initial conditions

The initial conditions of a mobile phone refer to finding a reference for further experiments. For experimental scenarios where manual inspection to the mobile device is possible, one immediate question is to find which processes are needed to keep running before the user starts an application manually.

In Android mobile devices such as the MotoE, the initial conditions refers to the state between when the Android operating system boot sequence has finished and before the user starts to use any application, which means the instance when the mobile is ready to interact with the user. This is technically the time after the Activity Manager starts the launcher application, which is the one that displays the home screen with all set icons.

From the process point of view, the initial conditions refer to list of all minimal processes that the mobile devices are executing after the boot sequence has finished. In this experiment, I have listed and analysed all processes of a smartphone MotoE XT1021 [7] after the boot sequence has finished. From this list, the child processes and native daemons triggered by the kernel, and the init process were required to keep running in order to offer functional service to the user. Listing 6.4 shows the console output of all processes that were not triggered directly by the init process (id 1) and kernel threads (id 2), it also excludes all "com." commercial applications.

```
%percy@xps:~\$ adb shell ps | awk '\$3!=1 {print \$0}' | awk '$3!=2 {print
%\$0}' %| grep -v com. | sort -k3 | nawk '{print NR"\t",\$0}'
```

[7] https://sites.google.com/site/meshawarenetworks/android/internals/finding-minimal-list-of-processes

```
LINE# USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
1     root      2     0     0      0     ffffffff 00000000 S kthreadd
2     root      1     0     832    628   ffffffff 00000000 S /init
3     radio     557   263   6156   1164  ffffffff 00000000 S
/system/bin/qmi_motext_hook
4     u0_a9     1185  265   336324 23324 ffffffff 00000000 S
android.process.acore
5     u0_a116   4006  265   344632 30436 ffffffff 00000000 S org.mozilla.firefox
6     system    1024  265   430392 54520 ffffffff 00000000 S system_server
7     root      298   287   4620   128   ffffffff 00000000 S daemonsu:master
8     root      2238  298   6732   300   ffffffff 00000000 S daemonsu:10093
9     root      3348  298   9808   232   ffffffff 00000000 S daemonsu:0
10     shell     3345  3339  460    4     c02415e4 000083c4 S su
11     root      3352  3348  8788   504   ffffffff 00000000 S daemonsu:0:3345
12     root      3507  3348  9812   532   ffffffff 00000000 S daemonsu:0:3504
13     root      3353  3352  0      0     ffffffff 00000000 Z daemonsu
14     root      3354  3352  936    484   ffffffff 00000000 S tmp-mksh
15     shell     3504  3498  460    4     c02415e4 000083c4 S su
16     root      3508  3507  0      0     ffffffff 00000000 Z daemonsu
17     root      3509  3507  932    476   ffffffff 00000000 S tmp-mksh
18     shell     4248  497   1248   236   00000000 b6edd498 R ps
19     shell     3498  497   932    460   c0108868 b6eee2e0 S /system/bin/sh
20     shell     3339  497   932    460   c0108868 b6f592e0 S /system/bin/sh
```

Listing 6.4: List of processes not triggered by init or kernel threads

From a manual analysis of List 6.4, the only process that could be killed is the "org.mozilla.firefox" as the rest of processes have a parent process pointing to the RIL (mediator between the phone service and the base band processor), init process, and adb shell daemon necessary for remote connections.

Finally, to list the processes that should be running at initial conditions of the MotoE XT 1021, it has been analysed whether the default applications installed in the MotoE belongs to the Google Android platform itself. The List 6.5 presents all commercial and organizational application for this smartphone.

```
%percy@xps:~\$ adb shell ps | awk '\$3!=1 {print \$0}' | awk '\$3!=2 {print
%\$0}' | grep -e 'com\.' -e 'org\.'

u0_a26    1096  265   392860 50200 ffffffff 00000000 S
com.android.systemui
u0_a68    1197  265   352636 28388 ffffffff 00000000 S
com.google.android.inputmethod.latin
u0_a21    1212  265   399912 41556 ffffffff 00000000 S
com.google.android.gms.persistent
radio     1235  265   368424 28692 ffffffff 00000000 S com.android.phone
system    1243  265   337048 24620 ffffffff 00000000 S
com.motorola.process.system
system    1250  265   336472 18676 ffffffff 00000000 S
com.qualcomm.services.location
```

```
u0_a73    1274  265   334568 18944 ffffffff 00000000 S com.motorola.modemservice
u0_a28    1288  265   486984 68740 ffffffff 00000000 S com.android.launcher
u0_a21    1316  265   404036 46512 ffffffff 00000000 S com.google.process.gapps
u0_a21    1409  265   330396 18328 ffffffff 00000000 S
com.google.process.location
u0_a21    1559  265   522044 49192 ffffffff 00000000 S com.google.android.gms
u0_a2     1669  265   329160 17260 ffffffff 00000000 S com.motorola.audioeffects
u0_a0     1718  265   357728 38056 ffffffff 00000000 S com.motorola.ccc
u0_a34    1992  265   348528 38712 ffffffff 00000000 S com.motorola.motocare
u0_a42    2023  265   378696 36684 ffffffff 00000000 S
com.google.android.googlequicksearchbox:search
u0_a36    3652  265   370740 36104 ffffffff 00000000 S com.android.vending
u0_a12    3709  265   330192 19348 ffffffff 00000000 S com.android.defcontainer
u0_a18    3761  265   340548 23400 ffffffff 00000000 S com.android.email
u0_a66    3994  265   380896 44188 ffffffff 00000000 S com.google.android.talk
u0_a116   4006  265   344632 33096 ffffffff 00000000 S org.mozilla.firefox
```

Listing 6.5: List of commercial and organizational applications for MotoE.

The conclusion of the individual analysis points that most of the commercial and organizational applications belong to Google Android and Motorola platform, and they usually appears on each time the MotoE is restarted. The only process that remains then is the "com.mozilla.firefox" package, which can be stopped manually.

By following this experimental procedure and by stopping all unnecessary processes after the boot sequence finished, such as the Firefox in our case, the initial conditions can be staged to start collecting energy data for the baseline.

6.4.1.2 Collecting energy measurements

The data collected for both stages, initial conditions and camera on, were under similar conditions. In both cases, the battery was fully charged before starting the experiment. Firstly, data for the stage in ”initial conditions” were collected. For the second stage, the Motorola camera application was started to record a video having the mobile phone in a static position. In both cases, the experiment was repeated the same number of times, utilising the same mobile and collecting the data during the same period of time.

The List 6.6 shows the format of the collected data in both stages and in all iterations. The first column represents the timestamps registered by the multimeter and the second one is the value of the current detected at that specific time in milliamperes. All experimental collected data[8] were cleaned using shell scripts and analysed using R.

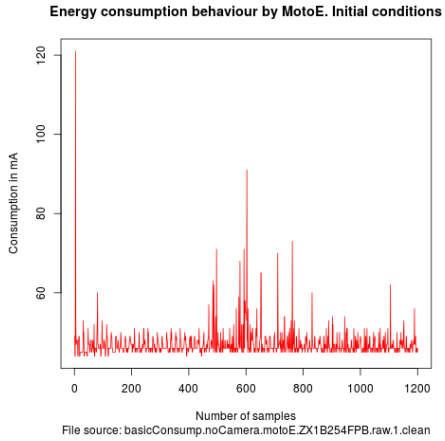
[8]https://bitbucket.org/percyperezd/jnfdexperiments/src/4f1199dee2e0/energy/basicconsumption/?at=master

```
TIME STAMP          CURRENT IN MILLIAMPERS
[2017-01-03 19:55:17.216] 000153?400:0
[2017-01-03 19:55:17.216] 000153?400:0
[2017-01-03 19:55:17.216] 000153?400:0
```

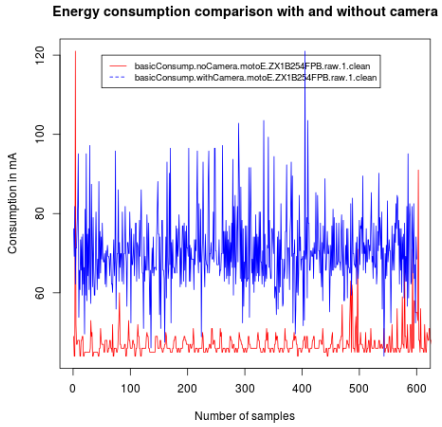
Listing 6.6: Sample of energy measurements using the UNI-T UT16E multimeter and minicom application.

6.4.1.3 Data analysis of results

Figure 6.21 shows the behaviour of the electric current flowing between the battery and the rest of the smartphone MotoE XT1021 when the mobile devices is set to its initial conditions and when the Motorola camera application is recording a video, both using the infrastructure described in Figure 6.20.



((a)) Initial conditions (baseline)



((b)) Comparison to camera recording video

Figure 6.21: Energy consumption for the baseline and the Motorola camera application

To quantify the energy consumed by the MotoE in its initial conditions (baseline), I estimated the mean across all measurements and also used the trapezoidal numeric integration to estimate the total area between the first and the last measurement. I utilised the following functions available in R to estimates two values: mean() and trapz(). The List 6.7 shows the estimations of the mean and the List 6.8 presents the overall areas of each iteration using these two R functions, where the means are in milliamperes(mA) and the areas in milliampere seconds (mAs).

```
# aggregate(dataset\$MILLIAMPERS,
list(dataset\$STAGE,dataset\$ITERATION),
mean)
STAGE   ITERATION  MILLIAMPERS
1    noCamera       1  46.96167
2  withCamera       1 217.88333
3    noCamera       2  47.55333
4  withCamera       2 221.95333
5    noCamera       3  47.12833
6  withCamera       3 211.84167
7    noCamera       4  47.51333
8  withCamera       4 217.19000
9    noCamera       5  48.11667
10 withCamera       5 220.43667
```

Listing 6.7: Overall means of energy consumption of the MotoE with Camera vs without Camera

```
# aggregate(dataset\$MILLIAMPERS,
list(dataset\$STAGE,dataset\$ITERATION),
function(x){trapz(c(1:length(x),x))})
STAGE   ITERATION  MILLIAMPERS/SEC
1    noCamera       1 208453.5
2  withCamera       1 310922.0
3    noCamera       2 208806.5
4  withCamera       2 313321.0
5    noCamera       3 208552.0
6  withCamera       3 307298.5
7    noCamera       4 208783.5
8  withCamera       4 310506.5
9    noCamera       5 209145.0
10 withCamera       5 312452.5
```

Listing 6.8: Estimated areas for energy consumption of the MotoE with Camera vs without Camera

Taking the means and areas as elements of an R list, the List 6.9 shows the estimations of overall means and areas, using the tapply() R function.

```
# Mean of means of all iterations in mA.

mean         sd     ymin      ymax
noCamera    47.45467 0.4474828  46.5776  48.33173
withCamera 217.86100 3.8750601 210.2659 225.45612

# Means of all areas of all iterations in mA per second
mean         sd     ymin     ymax
noCamera   208748.1  268.0661 208222.7 209273.5
withCamera 310900.1 2312.8784 306366.9 315433.3
```

Listing 6.9: Overall means and standard deviation for comparison: Camera vs without Camera

I utilised the values of the overall means and areas to estimate the difference between the baseline and another mobile application, such as the Motorola camera as in this case. Therefore, the baseline for the MotoE concludes that it consumes 57.99 mAh (208748.1/3600) and the overall electrical current that flows between the battery and the rest of the MotoE is 47.45 mA.

The energy consumed by the Motorola camera application is simply the arithmetic difference between the baseline and the energy consumed when the camera is recording a video, which means subtract the 57.99 mAh from the 86.36 mAh (310900.1/3600), that is 28.37 mAh. Finally, the electric current that flows between the battery and the rest of the mobile device when the

camera is recording a video is 170.41 mA (217.86100 -47.45467).

6.5 Validation of energy consumption measurements for data retrieval

This section presents experiments based on real networks of Android smartphones connected in ad-hoc mode. These small scale experiments aim to measure energy consumption from batteries in cases where a consumer request a content from a producer. The following sections consist of three experiments. The first one validates the energy consumption of retrieving content through JNFD and through a wireless Hotspot. Similarly, the second experiment validates JNFD and OLSR, and finally the last experiment validates the forwarding strategies of JNFD and wireless Hotspot. In all these three experiments the energy measurements on Android smartphones follow the procedure presented in Section 6.4.

6.5.1 Energy consumption of JNFD and wireless Hotspot

The purpose of this experiment is to measure energy consumption in an Android-based forwarder when a consumer retrieves content from a producer. This section presents the results of energy measurements between two approaches to retrieve data from a producer. The first approach utilises a JNFD forwarder to retrieve the content of interest from producer, while the second approach accomplishes the same objective by utilising wireless hotspot, instead of a JNFD forwarder. As a result, it is expected a lower overall energy consumption of JNFD compared with wireless hotspot.

6.5.1.1 Functional description

In the first case of this experiment, the only four available consumers were connected in ad-hoc mode to a forwarder, and this forwarder to a producer. The producer serves content of a file of 1.9 Mb, which is requested by the consumers. The consumer starts to send interest requests by pressing a start button located in the user interface windows of the JNFD application.

The consumers are started one after another and each consumer request the same file from the producer. This is because this experiments requires to also retrieve content from the forwarder's cache. The experiment starts when one of the consumers sends an interest request to the forwarder, and while the forwarder starts to satisfy this request by retrieving content from the producer, the other three consumers start to send in sequence the same request to the forwarder. The idea behind this sequence of requests from four consumers is to measure the overall energy

consumed by the forwarder when content is retrieved from the producer and from the forwarder's cache. The energy measurements are started to be collected just after the last consumer is triggered.

On the wireless hotspot case, the same Android smartphone selected as forwarder in the JNFD case is set into wireless hotspot mode. This allows the other smartphones to be connected and be able to reach each other. The consumer and the producer uses the same applications as in the case of JNFD. In comparison with JNFD, the only difference is that nodes are connected through a wireless hotspot instead of ad-hoc mode and content is retrieved, through the hotspot node, from the producer instead through a JNFD forwarder. The digital multimeter is connected to the smartphone that serves as hotspot. The experiment is then executed in a similar way as in the case of JNFD, consumers request sequentially the content of the file directly from the producer.

Once energy measurements on the forwarder are completed, the two cases of this experiment are repeated to measure energy consumption in the smartphone that acts as a producer, the digital multimeter is connected to this mobile phone. This experiment considers not necessary to connect the multimeter to each of the consumers due to the interest of this experiment is to measure energy consumption when content is retrieved from the producer and from the cache of the forwarder. Table 6.2 provides more details of the scenario of this experiment.

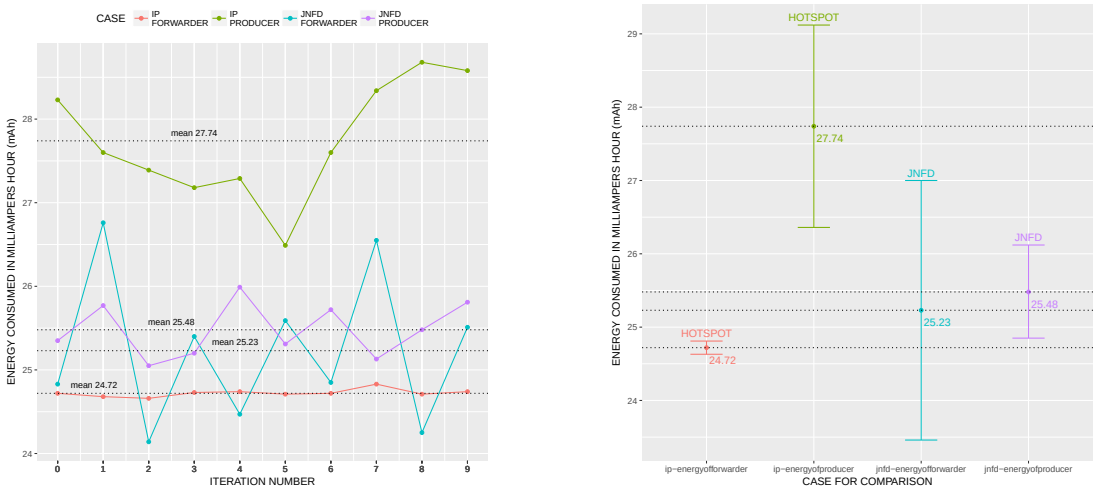
System Under Test	A forwarder running
System Parameters	Transport type: UDP Wi-Fi mode: adhoc and hotspot. Frequency: 2.412 GHz Tx power:20 dbm Wi-Fi Channel = 0 Screen of the forwarder = ON.
Environment Environment parameters	Topology: Four Consumers <-> Forwarder <-> Producer The forwarder executes JNFD and wireless IP hotspot. Wi-Fi mode: ad hoc, hotspot. Frequency: 2.412 GHz Tx power:20 dBm
Workload	Synthetic workload: The content loaded in the producer is a text file of 1.9 Mbytes, which corresponds to 500 segments of 4KB each.
Measurements	Cost: energy consumed in mAh by the forwarder when the consumers request an interest and the producer provides the respective content.

Table 6.2: Benchmarking scenario for energy consumption of a forwarder running JNFD and wireless hotpot

6.5.1.2 Data collection and analysis

I write shell scripts to extract the timestamps and the value of the measurements in milliamperes. These scripts also aggregate and organise the data in columns in the format that the R script can easily process and visualise.

Figure 6.22 depicts the statistical analysis of the collected data, which was generated by the R script. This figure has two parts: the first part is Figure 6.22(a), which plots the mean of energy consumption measurements on each of the iterations of the experiment for the cases when the producer and forwarder utilise JNFD and Hotspot. The second part is presented in Figure 6.22(b), which plots the mean of means, means represented on Figure 6.22(a), for both cases presented in Figure 6.22(a). This is in order to compare overall consumption between both cases, JNFD and Hotspot.



((a)) Means per iteration **((b))** Overall means (mean of means)

Figure 6.22: Energy consumption of producer and forwarder using JNFD and using Hotspot

The combination of the results in Figure 6.22 and the results from the TukeyHSD testing presented below show that there is enough evidence to accept the following two main comparisons. That the energy consumed by the JNFD forwarder is similar to the forwarder using IP base Hotspot (p value of 23.76%). However, the energy consumed by a producer utilising JNFD is lower than the producer connected utilising the Hotspot (p value of 0%).

```
Tukey multiple comparisons of means
95% family-wise confidence level

Fit: aov(formula = lm(round(aulgraph1\$MILLIAMPERS/3600, 2) ~ aulgraph1\$CASE))

      Case                   diff         lwr        upr
ip-energyofproducer    ip-energyofforwarder   3.014  2.29747615  3.7305239
jnfd-energyofforwarder ip-energyofforwarder   0.511 -0.20552385  1.2275239
jnfd-energyofproducer  ip-energyofforwarder   0.757  0.04047615  1.4735239
jnfd-energyofforwarder ip-energyofproducer   -2.503 -3.21952385 -1.7864761
jnfd-energyofproducer  ip-energyofproducer   -2.257 -2.97352385 -1.5404761
jnfd-energyofproducer  jnfd-energyofforwarder 0.246 -0.47052385  0.9625239

                        p adj
ip-energyofproducer    ip-energyofforwarder    0.0000000
jnfd-energyofforwarder ip-energyofforwarder    0.2375511
jnfd-Thisenergyofproducer  ip-energyofforwarder    0.0350325
jnfd-energyofforwarder ip-energyofproducer     0.0000000
jnfd-energyofproducer  ip-energyofproducer     0.0000000
jnfd-energyofproducer  jnfd-energyofforwarder  0.7918776
```

Listing 6.10: TukeyHSD test results for a forwarder and a producer using JNFD and Hotspot IP

To finalise the visualisations, Figure 6.23 shows the comparison of the total energy consumption of a producer and a forwarder using both JNFD and hotspot. As expected, the conclusion from this figure is that overall JNFD consumes less energy than using hotspot and by 1,75 mAh. This overall estimation represents totals of energy consumption from the producer and from the forwarder in both scenarios.

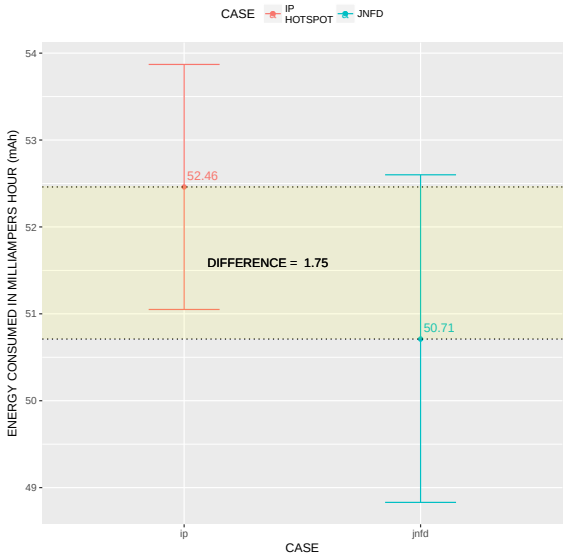
The next question is to see whether this difference in energy consumption between JNFD and hotspot is statistically significant. The List 6.11 presents the results of the TukeyHSD test which shows that there is enough evidence to accept that the overall consumption utilising JNFD is lower compared to the overall utilising hotspot (p value of 3,2%), and that the difference of 0.873 mAh is statistically significant.

```
Tukey multiple comparisons of means
95% family-wise confidence level

Fit: aov(formula = lm(round(aul\$x/3600, 2) ~ aul\$Group.1))

     diff      lwr         upr    p adj
jnfd-ip -0.873 -1.66705 -0.07894989 0.032045
```

Listing 6.11: TukeyHSD test results for JNFD versus Hotspot IP



((a)) Mean of means comparing JNFD and Hotspot IP

Figure 6.23: Total consumption of the producer and the forwarder using JNFD versus Hotspot. It includes intervals of confidence.

6.5.2 Energy consumption of JNFD and OLSR

This experiment has similar characteristics as the one where JNFD and wireless IP hotspot in Section 6.5.1. It validates the energy consumption of nodes that exchange information by using JNFD and OLSR. The main interest of this experiment is to estimate the overall energy consumption of a consumer-forwarder-producer scenario, in which the consumer retrieves the content of specific interest name stored in the producer. The expected results under the conditions of this experiment are statistically analysed to support that there is a significant overall difference between both approaches. This experiment is based on real mobile phones instead of simulations due to Mininet-WiFi does not support battery depletion models. A future work is to use the methodology of this section to generate a model that can be integrated in Mininet-WiFi.

6.5.2.1 Functional description

This experiment is divided into two parts: JNFD and OSLR. In both cases, nodes are linked in ad-hoc mode, as summarised in Table 6.3. In the JNFD part, a consumer, forwarder and producer utilise JNFD to retrieve a file of 1.9 MB. In this case, the consumer sends an interest packet to the forwarder to requests the content of a file. Then, the forwarder redirects the interest packet to the producer, who replies with the content of the name of interest. The content is re-transmitted by the forwarder until reaching the consumer. Due to the caching feature of JNFD, the forwarder

caches the content from the producer after the first request. Therefore, further requests of the same content can be satisfied by the forwarder's cache without the intervention of the producer, which means that the overall energy consumption is expected to be less than having a no cache, as it is the case of OLSR.

In the second part, OLSR, the three nodes route packets by using the OSLR daemon. In this scenario, the consumer has connectivity to the producer through the forwarder only since consumer and producer are not physically in range, therefore it is required that the forwarder redirect the packets between both nodes.

System Under Test	A forwarder and producer running JNFD and OLSR
System Parameters	Transport type: UDP Wi-Fi mode: ad-hoc and hotspot. Frequency: 2.412 GHz Tx power:20 dbm Wi-Fi Channel = 0 JNFD and OLSR service running in the background, no screen.
Environment Environment parameters	Topology: Four Consumers <-> Forwarder <-> Producer Nodes use OLSR and JNFD both in ad hoc mode. Wi-Fi link: ad-hoc. Frequency: 2.412 GHz Tx power:20 dBm Wi-Fi Channel = 0
Workload	Synthetic workload: The content loaded in the producer is a text file of 1.9 Mbytes, which corresponds to 500 segments of 4KB each.
Measurements	Cost: energy consumed in mAh by the forwarder when the consumers request an interest and the producer provides the respective content.

Table 6.3: Benchmarking scenario to compare energy consumption between forwarders running JNFD and OSLR in ad-hoc mode.

6.5.2.2 Data collection

The collected data is energy consumption measurements in the forwarder and producer using both cases: JNFD and OSLR. Additional measurements include cases in whether the content was cached or not by the forwarder. Table 6.4 summarises all these cases and shows that as the consumer has the same behaviour in all cases, energy measurements on this node are not required.

Part	Content cached in the forwarder?	Retrieving content
JNFD	NO (1st time only)	Consumer <–>Forwarder <–>Producer
	YES (from 2nd time)	Consumer <–>Forwarder
OLSR	NO(always)	Consumer <–>Forwarder <–>Producer

Table 6.4: Cases of energy consumption measurement for this experiment

Based on Table 6.4 the collected data sets are listed as follow:

- **1.- JNFD where content is not cached**. The cache of the forwarder does not contain the content requested by the consumer, therefore it is retrieved from the producer.
- **2.- JNFD with content cached**. The forwarder replies to the consumer with the interested content as the forwarder has previously cached it.
- **3.- OLSR no cache**. The forwarder redirects the packet between the consumer and the producer, and the interested content requested by the consumer is satisfied by the producer.

In all these three cases, the digital multimeter is connected to the forwarder and then to a producer, energy measurements are collected after the first segments of the content start to arrive at the consumer, and the collection of the data sets are stopped after four minutes. This procedure was repeated for each iteration and the collected data set stored in a file system for statistical analysis.

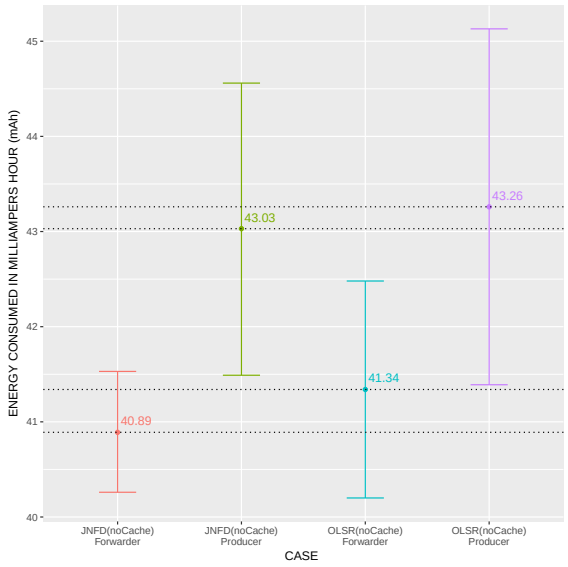
6.5.2.3 Data analysis

The collected data sets were cleaned and aggregated using shell scripts which are available in the JNFD experimental repository [9]. The statistical analysis was completed using R script and it is divided into three parts.

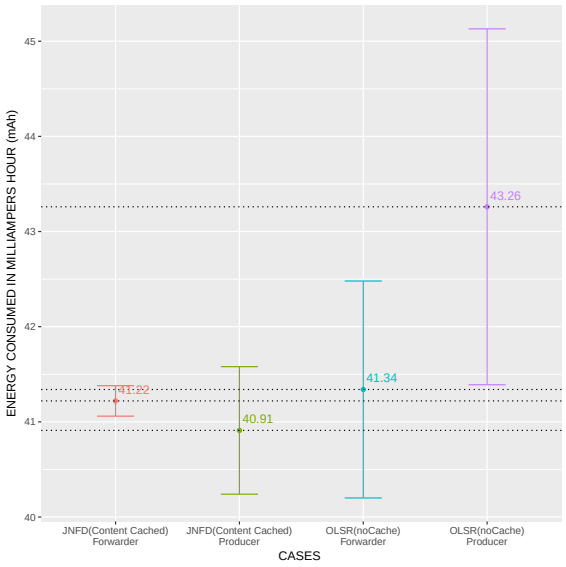
Part one The first part, which is depicted in Figure 6.24(a), validates JNFD and OLSR by estimating the difference of energy consumption between producers and forwarders when the content requested by the consumer is not cached in the forwarders, therefore for the case of JNFD forwarders redirect the request to the producers. In the case of OLSR, the forwarder routes the request the producer directly as it has not caching behaviour.

Figure 6.24(a) presents two types of validations: between forwarders and between producers. The results of the TukeyHSD test to compare forwarders, presented in List 6.12, shows that a OLSR forwarder consumes more energy than a JNFD forwarder. However, as the p value

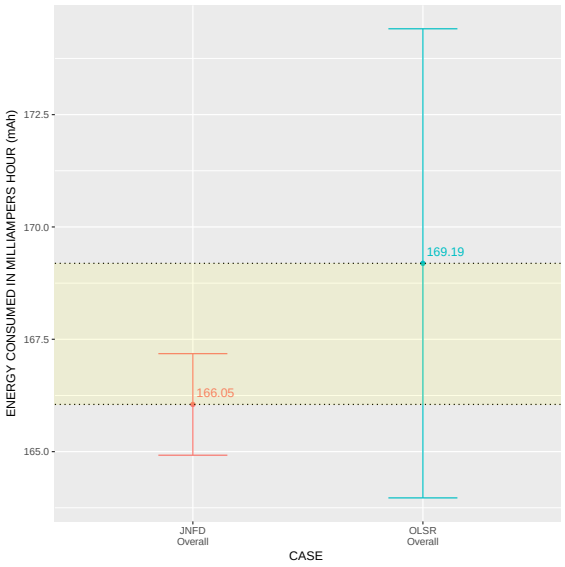
[9] https://percyperezd@bitbucket.org/percyperezd/jnfdexperiments.git



((**a**)) Content is not cached in the forwarder



((**b**)) Content is cached in the forwarder



((**c**)) Overall consumption comparison

Figure 6.24: Energy consumption measurements for JNFD and OLSR

is greater than 5%, this difference is not statistically significant. Consequently, based on the collected data set, there is enough evidence to accept that the energy consumption of the JNFD forwarder is similar to the energy consumption of the OLSR one, and that the difference of 0.44 mAh is not statistically significant.

```
  Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = lm(round(aulNoForw\$x/3600, 2) ~ aulNoForw\$Group.1))
diff        lwr      upr     p adj
olsrservice-jnfdservice 0.4433333 -0.1622761 1.048943 0.1339221
```

Listing 6.12: TukeyHSD test comparing JNFD and OLSR forwarders without caching

In the other hand, the test results of comparing producers are presented in List 6.13. Similarly to forwarders, even the OLSR producer consumed more energy than the JNFD one, this difference of 0.23 mAh is not statistically significant. The p value of 65.34% implies that based on the collected data, there is enough evidence to accept the hypothesis that the JNFD producer consumes similar energy as the OLSR one.

```
 Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = lm(round(aulNoProd\$x/3600, 2) ~ aulNoProd\$Group.1))
diff        lwr      upr     p adj
olsrservice-jnfdservice 0.2333333 -0.8899915 1.356658 0.6533998
```

Listing 6.13: TukeyHSD test comparing JNFD and OLSR producers without caching

Part two The second part is similar to the first one, with the difference that the JNFD forwarder has already cached the content requested by the consumer. Consequently, the producer does not receive any request from the forwarder. This behaviour does not occur in OLSR as the OLSR node does not provide a caching feature, the producer needs to reply all request from the consumer.

Figure 6.24(a) depicts the results of forwarders and producers, which are compared in the List 6.14 and List6.14. The results of a comparison of forwarders are presented in List 6.14. Even the OLSR node consumed more energy than the JNFD node, the difference of 0.115 mAh is not statistically significant, which is confirmed by the p-value of 64.19%. Therefore, based on the collected data sets, there is enough evidence the expected result, which is to accept that the energy consumption between the JNFD and OLSR node are similar.

```
 Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = lm(round(aulYesForw\$x/3600, 2) ~ aulYesForw\$Group.1))
diff        lwr       upr     p adj
olsrservice-jnfdservice 0.115 -0.4193844 0.6493844 0.6418987
```

Listing 6.14: TukeyHSD test comparing JNFD and OLSR forwarders with caching

However, comparing the JNFD and OSLR producers, the results of List 6.15 shows that the energy saved by the JNFD producer is statistically significant. This is due to the difference between both JNFD and OSLR producers has a p-value of almost zero percent, which confirms that the 2.35 mAh difference is significant.

```
  Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = lm(round(aulYesProd\$x/3600, 2) ~ aulYesProd\$Group.1))
diff     lwr     upr     p adj
olsrservice-jnfdservice 2.35 1.42736 3.27264 0.0002052
```

Listing 6.15: TukeyHSD test comparing JNFD and OLSR producers with caching

Part three The third part is depicted in Figure 6.24(c) and compares the overall energy consumption between JNFD and OLSR, which includes an aggregation of both previous parts. Considering the means of both cases, JNFD consumes less energy than OLSR by 3.14 mAh. The overall results of the TukeyHSD test presented in List 6.16 shows that there is enough evidence to accept that JNFD consumes less energy than OLSR, and that energy saved by JNFD is statistically significant (p-value of 1.81%).

```
 Tukey multiple comparisons of means
95% family-wise confidence level
Fit: aov(formula = lm(round(aulsum\$x/3600, 2) ~ aulsum\$Group.1))
diff       lwr      upr     p adj
olsrservice-jnfdservice 3.14 0.6597568 5.620243 0.0181347
```

Listing 6.16: TukeyHSD test comparing JNFD and OLSR overall energy consumption

6.5.3 Validation of data retrieval using JNFD forwarding strategies

This experiment aims to estimate the amount of energy consumed by a forwarder which uses each of the JNFD FIB next hop selection strategies. This experiment ranks these JNFD strategies by considering the level of energy consumption and efficiency. Considering the conditions of

this experiment, it is expected at the end of the data analysis of this experiment that forwarding strategies such as geo-location perform not as better as unicast when content is retrieved from a information producer.

6.5.3.1 Functional description

The Figure 6.25 shows the main infrastructure utilised to collect energy consumption measurements when a forwarder redirects packets using one of and each of the six available JNFD strategies. The mobile devices include three MotoE XT1021 smartphones, one Dell laptop with a Wi-Fi USB stick, a digital multimeter UT61E, and an infrared-RS232-USB cable. The roles that each component assumes in this experiment are described as following.

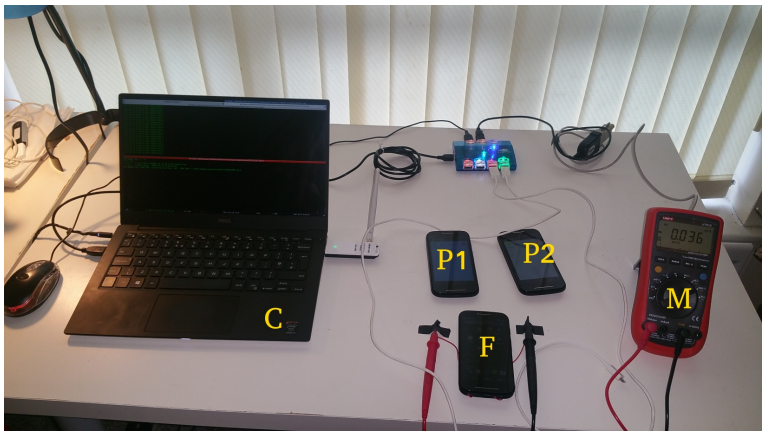


Figure 6.25: Infrastructure for comparison JNFD strategies and wireless hotspot

- The smartphone MotoE tagged with letter F acts as a JNFD forwarder. This forwarder is set manually to one specific strategy at a time by selecting the name of strategy from the menu on the user interface of JNFD.
- The smartphones MotoE tagged with letters P1 and P2 act as producer one and producer two. These producers store in their file systems a common pre-loaded 1.9 MB text file, which is equivalent to around 500 segments.
- The Dell laptop tagged with letter C has a Wi-Fi USB stick and acts as a consumer. This consumer is the node that requests the content of a name of interest from the producers.
- The multimeter tagged with letter M is connected to the forwarder. This multimeter collects the flow of energy charges between the battery and the rest of the MotoE smartphone .

Initial conditions Figure 6.26 depicts the initial conditions prior to data collections. Producer P1 and P2 register the prefix of the available files in the forwarder. Therefore, the producers set to a wait state for incoming requests, in particular from the forwarder.

On the other side, the consumer is set to request the same name registered by the producer. Consequently, the forwarder would be able to retrieve the content of the name of interest from one or both producers.

In the case of the wireless hotspot, it is required that the forwarder starts the wireless hotspot service to establish connectivity between the consumer and producers. Using this setting, the consumer requests the content of the file directly to the producer and not to the forwarder, the forwarder simply routes packets from one end to another.

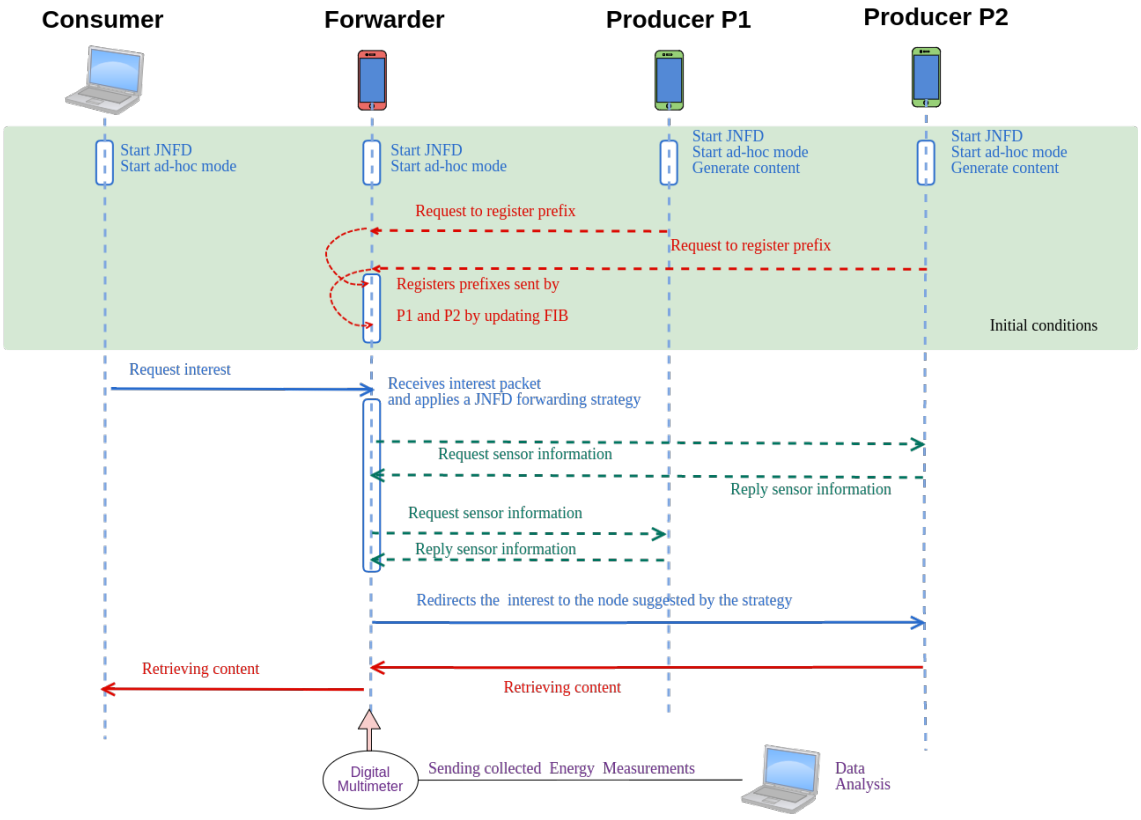


Figure 6.26: General sequence of steps for the experimentation

Functional description Besides the initial conditions, Figure 6.26 describes the general procedure followed in this experiment using one particular forwarding strategy at a time. The experiment begins when the consumer starts to send the 500 interest requests one after another, each interest request corresponds to one segment of the content of interest. Additionally, in the consumer, a performance file is created with the following metrics every time that an interest is

requested and a content is retrieved: the name of the interest, the timestamps in nanoseconds, the direction of the packet, and the size of the content of the data packet. The analysis of both energy consumption and performance files are explained in section 6.5.3.3.

Another detail is that the digital multimeter is always connected to forwarder and continuously measuring energy. The minicom application is utilised to start recording these measurements into a file just after the consumer is triggered. Table 6.5 summarises the scenario utilised to collect energy measurements consumed by a forwarder using one of FIB next hop selection strategies at a time.

System Under Test	JNFD forwarding strategies: baseline, unicast, random, geo-location, battery, storage, broadcast.
System Parameters	Transport type: UDP Wi-Fi mode: ad-hoc Frequency: 2.412 GHz Tx power: 20 dbm Wi-Fi Channel: 0 Screen of the forwarder : ON
Environment	Topology: Consumer <-> Forwarder <-> (Producer 1 , Producer 2) Forwarder selects on forwarder strategy at a time.
Environment parameters	Wi-Fi link: ad-hoc mode Frequency: 2.412 GHz Tx power:20 dBm Wi-Fi Channel = 0
Workload	Synthetic workload: a text file of 1.9 MBytes, 500 segments
Measurements	Cost: Energy consumed by the forwarder. Performance: Validity: Correctness, completeness Responsiveness: time to retrieve content.

Table 6.5: Benchmarking scenario for JNFD forwarding strategies

6.5.3.2 Data collection

Measurements begin to be collected after the initial conditions are set, and when the forwarder starts to receive requests from the consumer and satisfies them by retrieving the content from the producer. The data collected for this experiment consist of two types of logs stored in two different files per strategy: the energy consumed by the forwarder and the performance file.

- **Energy consumption file.** This file contains the amount of energy consumed by the forwarder during the experiment using a particular strategy. A clean file contains the time stamp and the number of milliamperes that the multimeter detects at that time. The multimeter records two measurements every second.

- **Performance file.** This file is generated by the consumer node, and it contains the time in nanoseconds when the interest has been sent and when its content was received, the size of the received content, and the requested name of interest .

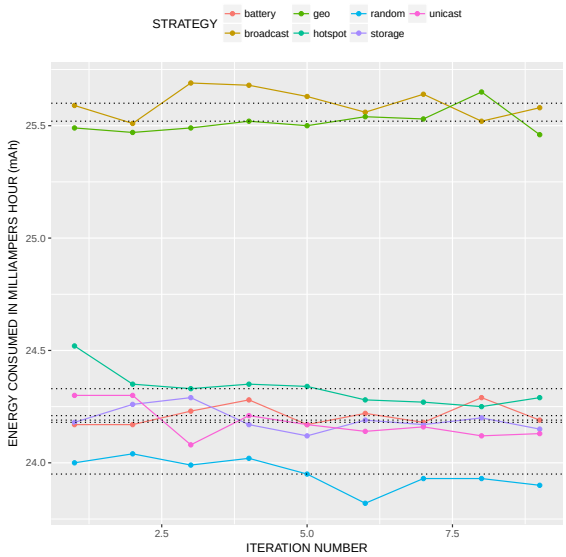
6.5.3.3 Data analysis

The statistical analysis of the collected data is divided into three parts, the first one analyses the energy consumption in the forwarder for each JNFD forwarding strategy, and the second one analyses its performance. Finally, the last part summarises previous two parts into to estimate the efficiency of the forwarder per strategy.

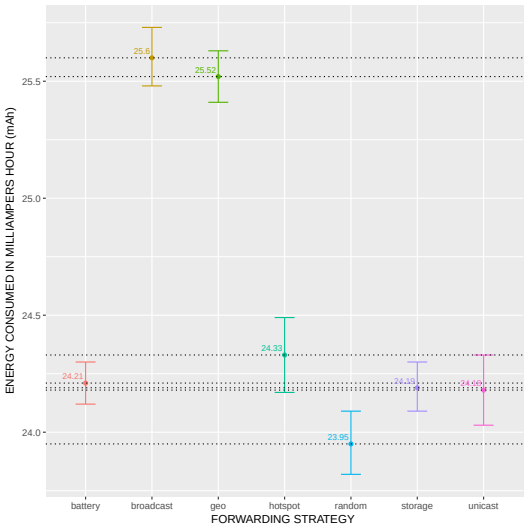
Energy consumption Figure 6.27(a) presents the means of the energy consumption measurements per forwarding strategy and per iteration, where an iteration correspond to run the experiment once. To estimate the differences in energy consumption between forwarding strategies, one of the methods is by calculating the means of each dataset in a similar way as it was described in section 6.4.1, which case compares the baseline and the energy consumed by the Motorola camera application. The results of these calculations are plotted in Figure 6.27(b), this figure plots the mean of means of the repetitions per forwarding strategy. Additionally, this figure also plots the interval of confidence associated to each mean of means.

Figure 6.28 ranks the forwarding strategies base on the value of the overall mean (mean of means of overall network traffic) showed in Figure 6.27(b). This figure presents to random strategy as the strategy that consumes less energy in this experiment, this is expected as random strategy does not make decisions between node status information from nearby nodes and randomly redirects the interest packet to one of the nearby nodes. This ranking places to battery, storage, unicast and random forwarding strategies with lower energy consumption than the traditional wireless hotspot. The immediate question is to check whether the difference in energy consumption between these forwarding strategies is significant.

Another method to calculate these means and statistically analyse whether their differences are significant is by using the TukeyHSD test, which results are presented in list 6.17.



((a)) Means per iteration



((b)) Overall consumption per iteration

Figure 6.27: Means and mean of means of energy consumption measurements per forwarding strategy

```
  Tukey multiple comparisons of means
95% family-wise confidence level

Fit: aov(formula = lm(round(aul\$MILLIAMPERS/3600, 2) ~ aul\$STRATEGY))

diff          lwr          upr     p adj
broadcast-battery  1.38888889  1.2953814  1.48239639 0.0000000
geo-battery        1.30555556  1.2120481  1.39906306 0.0000000
hotspot-battery    0.12000000  0.0264925  0.21350750 0.0042585
random-battery    -0.25777778 -0.3512853 -0.16427028 0.0000000
storage-battery   -0.01888889 -0.1123964  0.07461861 0.9959696
unicast-battery   -0.03222222 -0.1257297  0.06128528 0.9386617
geo-broadcast     -0.08333333 -0.1768408  0.01017417 0.1108375
hotspot-broadcast -1.26888889 -1.3623964 -1.17538139 0.0000000
random-broadcast  -1.64666667 -1.7401742 -1.55315917 0.0000000
storage-broadcast -1.40777778 -1.5012853 -1.31427028 0.0000000
unicast-broadcast -1.42111111 -1.5146186 -1.32760361 0.0000000
hotspot-geo       -1.18555556 -1.2790631 -1.09204806 0.0000000
random-geo        -1.56333333 -1.6568408 -1.46982583 0.0000000
storage-geo       -1.32444444 -1.4179519 -1.23093694 0.0000000
unicast-geo       -1.33777778 -1.4312853 -1.24427028 0.0000000
random-hotspot    -0.37777778 -0.4712853 -0.28427028 0.0000000
storage-hotspot   -0.13888889 -0.2323964 -0.04538139 0.0005712
unicast-hotspot   -0.15222222 -0.2457297 -0.05871472 0.0001267
storage-random     0.23888889  0.1453814  0.33239639 0.0000000
unicast-random     0.22555556  0.1320481  0.31906306 0.0000000
unicast-storage   -0.01333333 -0.1068408  0.08017417 0.9994308
```

Listing 6.17: TukeyHSD test results for a JNFD forwarding strategies comparisons

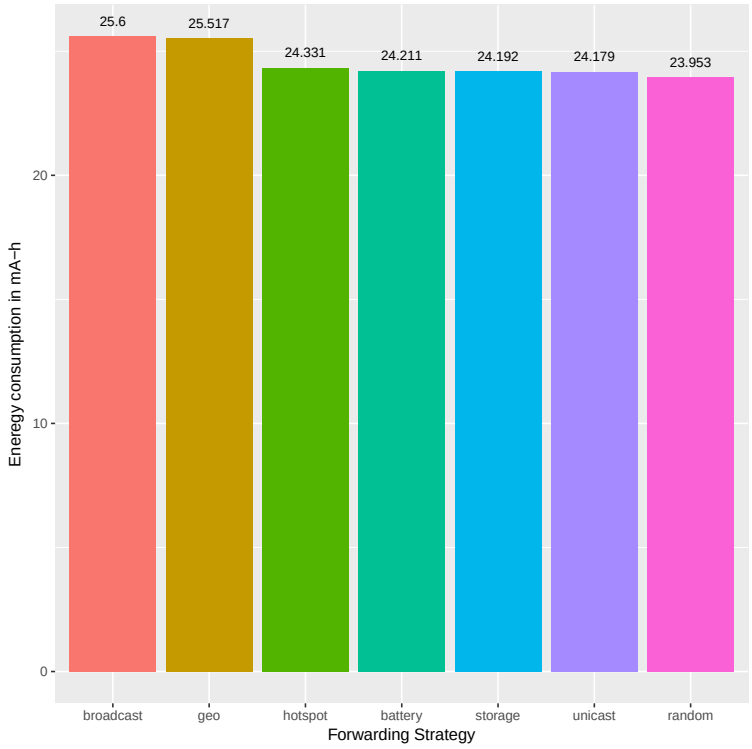


Figure 6.28: Rank of JNFD forwarding strategies base on their energy consumption

Considering 5% as p value for reference, the list 6.17 can be divided into two groups, one group of the comparisons where the p values are more than 5%, and the other where the p value is less than 5%. The List 6.18 presents the group of comparisons where the difference of energy consumption between pairs are less than 5%, which show enough evidence to reject the assumption that energy consumption of pairs is statistically similar. The second group is listed on List 6.19, which p values show that there is enough to accept the assumption that the difference of energy consumption of pairs is statistically similar. Both lists are ordered by the absolute value of the difference of the means.

```
Strategy comparison     Difference p-value
-------------------------------------------------

broadcast-battery  1.38888889  0.0000000
geo-battery        1.30555556  0.0000000
hotspot-battery    0.12000000  0.0042585
random-battery    -0.25777778  0.0000000
hotspot-broadcast -1.26888889  0.0000000
random-broadcast  -1.64666667  0.0000000
storage-broadcast -1.40777778  0.0000000
unicast-broadcast -1.42111111  0.0000000
hotspot-geo       -1.18555556  0.0000000
random-geo        -1.56333333  0.0000000
storage-geo       -1.32444444  0.0000000
unicast-geo       -1.33777778  0.0000000
random-hotspot    -0.37777778  0.0000000
storage-hotspot   -0.13888889  0.0005712
unicast-hotspot   -0.15222222  0.0001267
storage-random     0.23888889  0.0000000
unicast-random     0.22555556  0.0000000
```

Listing 6.18: Differences of overall means that are statistically significant, p<5%

```
Strategy comparison     Difference p-value
-------------------------------------------------

storage-battery   -0.01888889  0.9959696
unicast-battery   -0.03222222  0.9386617
geo-broadcast     -0.08333333  0.1108375
unicast-storage   -0.01333333  0.9994308
```

Listing 6.19: Differences of overall means that are not statistically significant,p>=5%

The combination of Figure 6.27 and List 6.19 show that forwarding strategies can be divided into two groups. The first group is formed by the storage, unicast, battery strategies, and the second group is formed by the geo-location and broadcast strategies. However, the energy consumed by both groups are statistically different. Note the GPS location were captured by using the internal GPS receiver on each mobile phone and through the Java API from Android platform.

On the other hand, the List 6.18 presents a list of statistical comparisons between strategies which energy consumption is different. Note that wireless hotspot consumes more energy than the battery, storage, unicast and random strategies, which is an evidence that by enhancing these type of JNFD forwarding strategies energy consumption can be reduced. As a conclusion, the final list of JNFD forwarding strategies can be grouped and ranked as in List 6.20.

```
broadcast ~ geo-location   ( high consume of energy)
hotspot
battery ~ storage ~ unicast
random               ( low consume of energy)
```

Listing 6.20: Ranking of JNFD forwarding strategies order by energy consumption

Efficiency

Responsiveness The List 6.21 shows the overall responsiveness per JNFD forwarding strategy. Responsiveness is understood as the estimated time in nanoseconds between the time when the interest has sent and the time when the content was received. Figure 6.29 depicts the times from List 6.21, and places to the unicast and random strategy with the highest responsiveness.

```
STRATEGY   DIFFTIMENANO  DIFFTIMESEC
unicast      136314475   0.1363145
random       127264265   0.1272643
broadcast    340710933   0.3407109
hotspot      506215357   0.5062154
storage      531225992   0.5312260
geo          642083694   0.6420837
battery      642224847   0.6422248
```

Listing 6.21: Overall means of the responsiveness

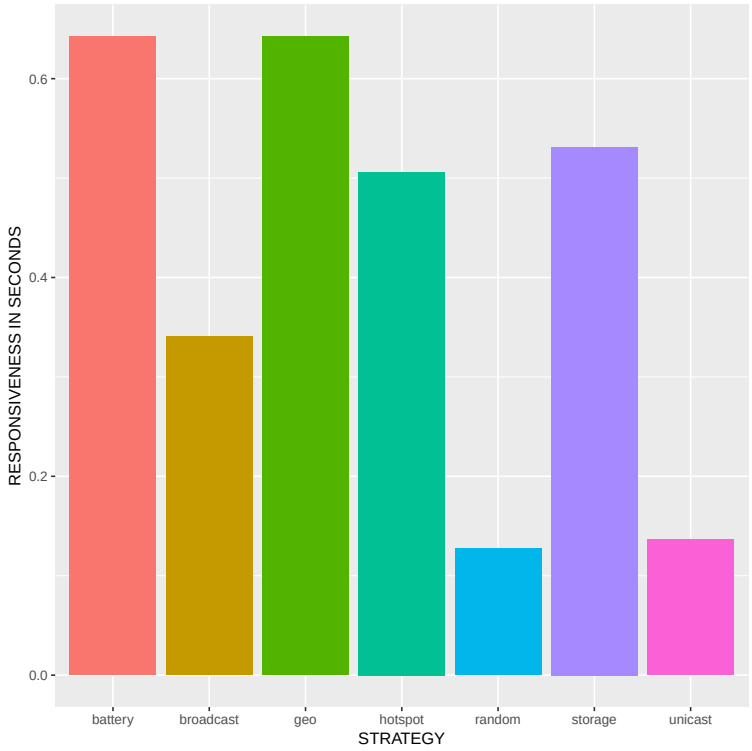


Figure 6.29: Responsiveness in seconds per strategy

Completeness To verify if each interest packet sent by the consumer has been replied with its respective data packet, the below pipe of shell commands read all clean files one at a time, extracts the sent and received names, and order them. By counting the number of unique occurrences is possible to detect which interest has more or less than two occurrences, one for the sent and another for received name. The results of executing this pipeline shell command show two cases. Firstly, the case when the number of occurrences is equal to two, this case can be understood as each request sent by the consumer was satisfied by the producer, one record for sending the request and another for receiving its content.

The second case is when the number of occurrences is equal to one, these cases were manually analysed and conclude that are packets received at the beginning and at the end of the experiment, which suggest that the experiment started and finished exactly after the interest was sent or just before when the data packet was received. These cases were discarded as they include only one of the occurrences and do not affect the final analysis.

```
percy@xps:clean\$ cat filterUnsatisfiyedInterest.sh
ls jnfd.*.filter1.clean | while read line;
do
 echo "--"\$line; nawk -F, '{ print \$6 }' \$line | nawk -F"/" '{ print \$3 }' | sort | uniq
 -c |
 nawk '\$1!=2{ print \$0 }'; echo "";
done | nawk '\$1!=1 {print \$0}' | grep -B 1 " c"
```

Listing 6.22: Command line to detect unsatisfied interest requests

I wrote a script that filters unsatisfied interest requests. The output of executing the shell command of List 6.22 is empty which shows that there are no inconsistencies, which can be understood as that all interests were replied with a data packet, which means a completeness of 100% for all JNFD strategies and wireless hotspot.

Correctness To verify that the content requested by the consumer corresponds to the replied by the producer, two comparisons were executed. In the first one, I scripted a comparison between the content received at the consumer and the one sent by the producer. The results of this comparisons were zero, which means that the ”diff” command did not find any differences. A second script compared the size of the content recorded by the consumer log. The performance files collected by the consumer include the name of the interest in the data packet and the size of the content field in the data packet. The Unix command line listed on 6.23 reads on each of the cleaned performance files and search for events where the size of the content field of the data packet is different to 4KB, which is the maximum size of the content field in an NDN data

packet.

```
percy@xps:clean\$ ls jnfd*.filter1.clean | while read line; do nawk -F, -v file=\$line '\$5=="
 received" && \$8!=4000 { print \$0","file }' \$line; done
```

Listing 6.23: Command line to detect a received packet size different than 4KB

The console output of List 6.23 shows that all packets have arrived with the correct size in the content field, consequently, this scripts concludes that the correctness in each of the strategies is 100%.

6.5.3.4 Summary of results

Merging all results from precedent subsections, Table 6.6 summarizes all of them in on single table. Note that correctness and completeness is all cases are 100%.

System under test	Cost-Energy [mA-h]	Responsiveness [seconds]
broadcast	25.60	0.34
geo	25.52	0.64
hotspot	24.33	0.51
battery	24.21	0.64
storage	24.19	0.53
unicast	24.18	0.14
random	23.95	0.13

Table 6.6: Table for benchmarking of the results

As efficiency is estimated arithmetically as the rate of a performance metric against the cost (performance/cost), which means that efficiency has direct relation with performance, and inverse relation with costs, as follow.

$$\textbf{Efficiency of Strategy X} = \frac{\textbf{Performance Metric of X}}{\textbf{Cost of strategy X}}$$

In the following paragraphs, two cases are analysed. The first one refers to estimate efficiency base on the percentage of correctness and completeness, and the second one refers to estimate efficiency base on the percentage of responsiveness. In both cases, the cost is represented by the energy consumption associated with a forwarding strategy.

Considering the performance metric of Table 6.6, all packets arrived to the consumer correctly and completely, which means that for these two performance metrics, the efficiency of a JNFD forwarding strategy is defined by its energy consumption. In other words, for the cases where all packets arrived correctly and completely, the less energy a forwarder strategy consumes, the more efficient. Therefore, the ranking presented on List 6.28 also represents the ranking of efficiency base on correctness and completeness. From this list, it is possible to say that in this group, random is the most efficient forwarding strategy.

For the case of responsiveness, the below formula describes its relationship with performance. Responsiveness has inverse relation to performance, it represents how fast the interest of the consumer is satisfied, which means the faster the content arrives at the consumer the better its performance.

$$\textbf{Performance Metric of X} = \frac{1}{\textbf{Responsiveness}}$$

Consequently, considering of responsiveness as a metric for performance, a JNFD forwarding strategy has high efficiency in the case that it has low responsiveness and low energy consumption, as it is represented in the next formula.

$$\textbf{Efficiency of Strategy X} = \frac{1}{\textbf{Responsiveness} * \textbf{Cost of strategy X}}$$

Table 6.7 shows quantitatively how the efficiency of each of the JNFD strategies is calculated for responsiveness. Note that the efficiency of the broadcast forwarding strategy is higher than a hotspot. This is due to broadcast responsiveness is faster than hotspot. To finalise, the most efficient JNFD forwarding strategies include random, and unicast, and the less efficient forwarding strategies include geo-location, battery, and storage.

6.6 Chapter summary

This chapter presents the validation of the nMANET framework through a set of experiments using Android smartphones and simulations of ad-hoc networks. In general, each of the experiments includes a functional description of the objective of the experiment, the methodology utilised to collect data and the analysis of the collected experimental datasets, which supports the

System under test	Cost-Energy [mA-h]	Responsiveness [seconds]	Efficiency
broadcast	25.60	0.34	1/(25.60)*1/(0.34)= 0.114
geo-location	25.52	0.64	1/(25.52)*1/(0.64)= 0.060
hotspot	24.33	0.51	1/(24.33)*1/(0.51)= 0.080
battery	24.21	0.64	1/(24.21)*1/(0.64)= 0.064
storage	24.19	0.53	1/(24.19)*1/(0.53)= 0.077
unicast	24.18	0.14	1/(24.18)*1/(0.14)= 0.292
random	23.95	0.13	1/(23.95)*1/(0.13)= 0.315

Table 6.7: Benchmarking of efficiency sbased on the responsiveness of a forwarding strategy

expected results.

The results of the experiments presented in this chapter show the empirical evidence of this thesis. The two main groups include experimentation in Mini-JNFD and in real Android smartphones. The methodology of experimentation and analysis of the collected data are validates the nMANET framework presented in this chapter. The results follow an analysis of JNFD and existing technologies, such as HTTP, OLSR and Wireless hotspot, and support the validation of all experiments. Finally, the discussions presented in this chapter enrich the conclusions presented in Chapter 7.

CHAPTER SEVEN

CONCLUSIONS

This chapter presents overall conclusions derived from previous chapters and their relationships. The content presented in previous chapters demonstrates that nMANET can enable responsible research on applying named data networking in MANETs through reproducibility, which was the main research challenge presented in Chapter 1. To provide evidence that nMANET can address this challenge, this thesis presents its design, in Chapter 4, and the development of the JNFD prototype, in Chapter 5. Additionally, the experimental framework of JNFD was needed in order to generate empirical data to validate nMANET experiments in virtual and real networks. The value of this empirical evidence resides in its reproducibility and the validation of expected results deduced from the data analysis.

The software development of nMANET follows the principles of continuous delivery and continuous integration practices. JNFD and its testing and experimentation framework can be applied in a wider context of experiments and future new scenarios, such as the cases presented in Chapter 6. The results from experiments in Chapter 6 support the conclusions of this thesis, and present the evidence that nMANET offers all relevant academic and technical content and tools to generate reproducible evidence for further experimentation with NDN in MANETs.

The conclusions from the analysis of the empirical data generated on each of the experiment, in Chapter 6, support and validate the expected outcome described at the beginning of each experiment. The methodology for validation presented in this chapter covers all main stages for reproducibility in empirical research. First, it covers the description of the designed experiment and its expectations, prior to analysis of the collected experimental data that may or may not support and validate the expected behaviour. Further, as the testing and experimentation framework of nMANET facilitates an environment that allows researchers to reproduce the validation of experiments, the framework of nMANET enables responsible research into

the application of NDN in MANETs. Note that researchers can analyse and reproduce the experimental data to validate the expected behaviour of an experiment.

Thought the implementations of the JNFD testing framework and the results from experiments using the nMANET framework, this thesis has shown the practical feasibility of JNFD in Android and Linux-based mobile devices and in virtual networks through an adaptation of the Mininet-WiFi simulator. This is important to highlight as NDN cannot be applied directly to MANETs; NDN was neither designed for traditional wireless networks nor MANETs. As a contribution, nMANET was created to facilitate the use of name-based communication in mobile ad-hoc networks.

The information provided in this thesis contributes to the NDN community with empirical evidence that shows the benefits of name-based networking in MANETs. Experiments used simulations to validate network traffic measurements, and also used real mobile devices to validate energy consumption from batteries of smartphones. Note that it is important to understand these results and conclusions in a wider context, as they open up more and new research questions and challenges, which require future work.

Chapter 2 and Chapter 3 present the limitations of exiting ICN approaches applied in MANETs for data distribution, and the need to provide to the community a framework to enable responsible research of ICN instances (such as NDN) in MANETs. The nMANET framework fulfils this need and ensures reproducibility and validation of experiments in a responsible manner.

7.1 Future work

Due to the multi-faceted nature of the research area of this thesis, there is a wide avenue for future work and for extending the nMANET approach in different directions, some of which are presented as follows.

Security The basis for security in an NDN network is a public-key infrastructure that ensures the integrity of the content. nMANET would require the same infrastructure to be in place that would allow consumers to verify the signatures on data packets. The creation of such an infrastructure is beyond the scope of this thesis. Ad-hoc networks are, by their nature, open to new nodes and so security is not provided at the point of network access. Any sensitive information would therefore need to be encrypted as well as signed.

New resource-aware forwarding strategies Design, deployment, testing and experimentation of new forwarding strategies is an open research area that JNFD and its testing and experimen-

tation framework can support. New forwarding strategies are required to efficiently use the available resources in mobile devices and also information provided by the available sensors in new mobile devices.

In addition, the mechanisms supporting forwarding strategies such as node status packets need to be further developed. In larger deployments, the size of the FIB entries will exceed the capacity of a single node status packet and decisions will need to be made as to what information to include. More effective ways to limit the information required in FIBs will need to be developed as well as ways to incrementally transfer FIB information.

Cancelling interests The prefix discovery process potentially establishes multiple reverse paths along which data can travel back to the consumer. This can lead to duplicate and unnecessary data transfers, which can potentially be avoided if a forwarder or consumer can declare to its neighbours that an interest has been fulfilled. This is especially important for strategies such as unicast or broadcast that activate multiple next hops.

Scaling JNFD needs to be tested in medium and large Android smartphone networks in order to replicate and contrast the experiments simulated through JNFD. These field tests and experiments are also important as they can generate mobility traces that can be used in future simulations. Additionally, these recorded mobility traces can be used to propose mobility models that model behaviour in emergency situations more realistically.

Raw socket communication While JNFD works well on top of UDP, additional efficiency gains are possible if it is instead deployed directly using frames. This should be relatively easy to do with the implementation of a new `Transport` implementation once NDN defines a mapping to raw Ethernet. One problem that remains, though, is that the Android Java SDK does not provide the necessary access to the underlying network layers.

Energy consumption At the time of this writing, Mininet-WiFi does not allow nodes to set the initial battery level or set a depletion model that can be simulated during the experiment. One direction for future work in this area therefore consists of incorporating Android-based energy consumption measurements into a depletion model that can be used in Mininet-WiFi or other simulators.

Streaming data nMANET easily supports the transport of multimedia data such as recorded videos, but its use for real-time streaming data such as for voice communication has not been explored, as the focus of this thesis is on content distribution.

Alternative to root access permissions Currently, nMANET works in ad-hoc mode, which requires root access on Android devices. This requirement means that users have to unlock their phones' bootloaders and install a utility such as `sudo`. This is a limitation of this project, and further alternatives need to be explored. One such alternative is WiFi-Direct, which is better supported by device manufacturers.

Real field trials JNFD as a prototype was not tested in real scenarios or emergency situations. The resource requirements for such a test to be conducted in a systematic way are high and also require a further refinement of the Android user interface. Future work is desirable to execute tests and experiments within a small community of 10 or 20 people over the course of a number of hours, simulating an emergency situation. Such a field trial may need to be run repeatedly to explore design alternatives as well as to control variables that influence the outcomes.

REFERENCES

[Abolhasan et al., 2009] Abolhasan, M., Hagelstein, B., and Wang, J.-P. (2009). Real-world performance of current proactive multi-hop mesh protocols. In *15th Asia-Pacific Conference on Communications, 2009. APCC 2009.*, pages 44–47. IEEE.

[Afanasyev et al., 2016] Afanasyev, A., Halderman, J. A., Ruoti, S., Seamons, K., Yu, Y., Zappala, D., and Zhang, L. (2016). Content-based security for the web. In *Proceedings of the 2016 New Security Paradigms Workshop*, pages 49–60. ACM.

[Ahlgren et al., 2012] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine*, 50(7).

[Al-Akkad and Zimmermann, 2011] Al-Akkad, A. and Zimmermann, A. (2011). User study: Involving civilians by smart phones during emergency situations. In *Proceedings of the 8th International ISCRAM Conference*, Lisbon, Portugal.

[Anjum et al., 2015] Anjum, S. S., Noor, R. M., and Anisi, M. H. (2015). Survey on manet based communication scenarios for search and rescue operations. In *2015 5th International Conference on IT Convergence and Security (ICITCS)*, pages 1–5. IEEE.

[Atkinson, 2005] Atkinson, R. (2005). An overview of the identifier-locator network protocol (ILNP). Research Note RN/05/22, University College London.

[Atkinson and Bhatti, 2012] Atkinson, R. and Bhatti, S. (2012). Identifier-locator network protocol (ILNP) architectural description. Request for Comments RFC 6740, Internet Research Task Force.

[Atkinson et al., 2009] Atkinson, R., Bhatti, S., and Hailes, S. (2009). ILNP: mobility, multi-homing, localised addressing and security through naming. *Telecommunication Systems*, 42(3):273–291.

[Baccelli et al., 2014] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, T. C., and Wählisch, M. (2014). Information centric networking in the iot: Experiments with ndn in the wild. In

Proceedings of the 1st International Conference on Information-centric Networking, pages 77–86. ACM.

[Bari et al., 2012] Bari, M. F., Chowdhury, S. R., Ahmed, R., Boutaba, R., and Mathieu, B. (2012). A survey of naming and routing in information-centric networks. *IEEE Communications Magazine*, 50(12).

[Bhatti et al., 2011] Bhatti, S. N., Atkinson, R., and Klemets, J. (2011). Integrating challenged networks. In *Military Communications Conference, 2011-MILCOM 2011*, pages 1926–1933. IEEE.

[Boukerche et al., 2011] Boukerche, A., Turgut, B., Aydin, N., Ahmad, M. Z., Bölöni, L., and Turgut, D. (2011). Routing protocols in ad hoc networks: A survey. *Computer networks*, 55(13):3032–3080.

[Camp Tracy and Vanessa, 2002] Camp Tracy, B. J. and Vanessa, D. (2002). A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502.

[Clausen and Jacquet, 2003] Clausen, T. and Jacquet, P. (2003). Optimized link state routing protocol (OLSR). Request for Comments RFC 3626, IETF Network Working Group.

[Collberg et al., 2015] Collberg, C., Proebsting, T., and Warren, A. M. (2015). Repeatability and benefaction in computer systems research. Technical Report TR 14-04, University of Arizona.

[Compagno et al., 2015] Compagno, A., Conti, M., Ghali, C., and Tsudik, G. (2015). To nack or not to nack? negative acknowledgments in information-centric networking. In *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*, pages 1–10. IEEE.

[Crowley and Chan, 2011] Crowley, J. and Chan, J. (2011). Disaster relief 2.0: The future of information sharing in humanitarian emergencies. Technical report, UN Foundation & Vodafone Foundation Technology Partnership, Washington, DC and Berkshire, UK.

[Dannewitz et al., 2013] Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., and Karl, H. (2013). Network of information (netinf)–an information-centric networking architecture. *Computer Communications*, 36(7):721–735.

[Drummond and Crawford, 2014] Drummond, J. and Crawford, N. (2014). Humanitarian crises, emergency preparedness and response: the role of business and the private sector: Kenya case study. Technical report, Overseas Development Institute.

[Edens and Scott, 2017] Edens, G. and Scott, G. (2017). The packet protector. *IEEE Spectrum*, 54(4):42–48.

[Effelsberg et al., 2013] Effelsberg, W., Steinmetz, R., and Lehn, M. (2013). Introduction. In Effelsberg, W., Steinmetz, R., and Strufe, T., editors, *Benchmarking Peer-to-Peer Systems*. Springer.

[Fontes et al., 2015] Fontes, R. R., Afzal, S., Brito, S. H., Santos, M. A., and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 384–389. IEEE.

[Freire et al., 2016] Freire, J., Fuhr, N., and Rauber, A. (2016). Reproducibility of data-oriented experiments in e-science (dagstuhl seminar 16041). In *Dagstuhl Reports*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Friginal et al., 2011] Friginal, J., De Andres, D., Ruiz, J.-C., and Gil, P. (2011). Towards benchmarking routing protocols in wireless mesh networks. *Ad Hoc Networks*, 9(8):1374–1388.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

[Gardner-Stephen et al., 2017] Gardner-Stephen, P., Challans, R., Lakeman, J., Bettison, A., Lieser, P., Steinmetz, R., Alvarez, F., and Lloyd, M. (2017). Productizing humanitarian telecommunications research: A case study of the serval mesh extender. In *Global Humanitarian Technology Conference (GHTC), 2017 IEEE*, pages 1–10. IEEE.

[Gardner-Stephen and Palaniswamy, 2011] Gardner-Stephen, P. and Palaniswamy, S. (2011). Serval mesh software-wifi multi model management. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, pages 71–77. ACM.

[Ghodsi et al., 2011] Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B., and Wilcox, J. (2011). Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM.

[Gray et al., 2004] Gray, R. S., Kotz, D., Newport, C., Dubrovsky, N., Fiske, A., Liu, J., Masone, C., McGrath, S., and Yuan, Y. (2004). Outdoor experimental comparison of four ad hoc routing algorithms. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '04, pages 220–229, New York, NY, USA. ACM.

[Haas, 1997] Haas, Z. J. (1997). A new routing protocol for the reconfigurable wireless networks. In *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*, volume 2, pages 562–566 vol.2.

[Hong et al., 2013] Hong, S., Jang, M.-W., and Lee, B.-J. (2013). Ccn networking architecture for mobile applications. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 609–612. IEEE.

[Hong et al., 1999] Hong, X., Gerla, M., Pei, G., and Chiang, C.-C. (1999). A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60. ACM.

[Hoque et al., 2013] Hoque, A., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., and Wang, L. (2013). NLSR: named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 15–20. ACM.

[Hsu et al., 2007] Hsu, W.-J., Spyropoulos, T., Psounis, K., and Helmy, A. (2007). Modeling time-variant user mobility in wireless mobile networks. In *INFOCOM 2007. 26th IEEE international conference on computer communications. IEEE*, pages 758–766. IEEE.

[Hughes and Palen, 2009] Hughes, A. L. and Palen, L. (2009). Twitter adoption and use in mass convergence and emergency events. *International Journal of Emergency Management*, 6(3-4):248–260.

[Humble and Farley, 2010] Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

[ITU, 2017a] ITU (2017a). Global and regional ICT data.

[ITU, 2017b] ITU (2017b). *Measuring the Information Society Report 2017, Volume 1*. International Telecommunication Union, Geneva Switzerland.

[Jacobson et al., 2009a] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009a). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM.

[Jacobson et al., 2009b] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009b). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM.

[Jacquet et al., 2001] Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. In *Proceedings of IEEE INMIC*.

[Jiang and Bi, 2014] Jiang, X. and Bi, J. (2014). ncdn: Cdn enhanced with ndn. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 440–445. IEEE.

[Johnson, 2013] Johnson, C. (2013). *Focus on technology and the future of humanitarian action*.

[Kaye et al., 2012] Kaye, J., Holstius, D., Seto, E., Eddy, B., and Ritter, M. (2012). Using nfc phones to track water purification in haiti. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 677–690. ACM.

[Koponen et al., 2007] Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 181–192. ACM.

[Kurkowski et al., 2005] Kurkowski, S., Camp, T., and Colagrosso, M. (2005). Manet simulation studies: the incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61.

[Kurose and Ross, 2013] Kurose, J. F. and Ross, K. W. (2013). *Computer networking: a top-down approach*. Addison Wesley.

[Lane et al., 2010] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9).

[Lantz et al., 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

[Lehman et al., 2016] Lehman, V., Hoque, A., Yu, Y., Wang, L., Zhang, B., and Zhang, L. (2016). A secure link state routing protocol for ndn. Technical Report NDN-0037, NDN Project.

[Lien et al., 2009] Lien, Y. N., Jang, H. C., and Tsai, T. C. (2009). A manet based emergency communication and information system for catastrophic natural disasters. In *29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 412–417.

[Loo et al., 2011] Loo, J. H., Mauri, J. L., and Ortiz, J. (2011). *Mobile ad hoc networks: current status and future trends*. CRC Press.

[Ma et al., 2014] Ma, G., Chen, Z., Cao, J., Guo, Z., Jiang, Y., and Guo, X. (2014). A tentative comparison on cdn and ndn. In *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2893–2898. IEEE.

[Mahadevan, 2014] Mahadevan, P. (2014). Ccnx 1.0 tutorial. *Technical Report, Xerox Palo Alto Research Center-PARC*.

[Marczyk et al., 2005] Marczyk, G., DeMatteo, D., and Festinger, D. (2005). *Essentials of research design and methodology.* John Wiley & Sons Inc.

[Meier, 2015] Meier, P. (2015). *Digital humanitarians: how big data is changing the face of humanitarian response*. Crc Press.

[Meisel et al., 2010a] Meisel, M., Pappas, V., and Zhang, L. (2010a). Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8. ACM.

[Meisel et al., 2010b] Meisel, M., Pappas, V., and Zhang, L. (2010b). Listen first, broadcast later: Topology-agnostic forwarding under high dynamics. In *Annual conference of international technology alliance in network and information science*, page 8.

[Ming et al., 2014] Ming, Z., Xu, M., and Wang, D. (2014). Age-based cooperative caching in information-centric networking. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE.

[Mishra and Pattanayak, 2013] Mishra, S. and Pattanayak, B. K. (2013). Power aware routing in mobile ad hoc networks-a survey. *ARPN Journal of Engineering and Applied Sciences*, 8(3).

[Mosko et al., 2014a] Mosko, M., Solis, I., Mahadevan, P., and Uzun, E. (2014a). CCNx 1.0 Naming: Transforming Network Addresses to Application Value.

[Mosko et al., 2014b] Mosko, M., Solis, I., and Uzun, E. (2014b). CCN 1.0 Protocol Architecture.

[Mosko et al., 2015] Mosko, M., Solis, I., Uzun, E., and Wood, C. (2015). Ccnx 1.0 protocol architecture. *Tech. Rep.*

[Murthy and Manoj, 2004] Murthy, C. S. R. and Manoj, B. (2004). *Ad hoc wireless networks: Architectures and protocols*. Prentice Hall.

[Nugroho et al., 2014] Nugroho, E., Sahroni, A., et al. (2014). Zigbee and wifi network interface on wireless sensor networks. In *Electrical Engineering and Informatics (MICEEI), 2014 Makassar International Conference on*, pages 54–58. IEEE.

[Nygren et al., 2010] Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The Akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19.

[Oh et al., 2010] Oh, S. Y., Lau, D., and Gerla, M. (2010). Content centric networking in tactical and emergency manets. In *Wireless Days (WD), 2010 IFIP*, pages 1–5. IEEE.

[Pan et al., 2011] Pan, J., Paul, S., and Jain, R. (2011). A survey of the research on future internet architectures. *IEEE Communications Magazine*, 49(7).

[Papadopoulos et al., 2010] Papadopoulos, F., Krioukov, D., Boguñá, M., and Vahdat, A. (2010). Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

[Perino and Varvello, 2011] Perino, D. and Varvello, M. (2011). A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 44–49. ACM.

[Perkins et al., 2003] Perkins, C., Belding-Royer, E., and Das, S. (2003). Ad hoc on-demand distance vector (aodv) routing. Request for Comments RFC 3561, IETF Network Working Group.

[Rhee et al., 2011] Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. J., and Chong, S. (2011). On the levy-walk nature of human mobility. *IEEE/ACM transactions on networking (TON)*, 19(3):630–643.

[Roy, 2010] Roy, R. R. (2010). *Handbook of mobile ad hoc networks for mobility models*. Springer Science & Business Media.

[Royer and Toh, 1999] Royer, E. M. and Toh, C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55.

[Saroiu et al., 2002] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. (2002). An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36(SI):315–327.

[Seskar et al., 2011] Seskar, I., Nagaraja, K., Nelson, S., and Raychaudhuri, D. (2011). Mobility-first future internet architecture project. In *Proceedings of the 7th Asian Internet Engineering Conference*, pages 1–3. ACM.

[Shafiq et al., 2014] Shafiq, M. Z., Liu, A. X., and Khakpour, A. R. (2014). Revisiting caching in content delivery networks. In *ACM SIGMETRICS Performance Evaluation Review*, volume 42, pages 567–568. ACM.

[Shi et al., 2017] Shi, J., Newberry, E., and Zhang, B. (2017). On broadcast-based self-learning in named data networking. In *IFIP Networking*.

[Sivasubramanian et al., 2004] Sivasubramanian, S., Szymaniak, M., Pierre, G., and Steen, M. v. (2004). Replication for web hosting systems. *ACM Computing Surveys (CSUR)*, 36(3):291–334.

[Smetters and Jacobson, 2009] Smetters, D. and Jacobson, V. (2009). Securing network content. Technical report, Technical report, PARC.

[Sommerville, 2010] Sommerville, I. (2010). *Software Engineering*. Addison-Wesley.

[Thomas et al., 2012] Thomas, J., Robble, J., and Modly, N. (2012). Off grid communications with android meshing the mobile world. In *IEEE Conference on Technologies for Homeland Security (HST)*, pages 401–405. IEEE.

[Trossen et al., 2010] Trossen, D., Sarela, M., and Sollins, K. (2010). Arguments for an information-centric internetworking architecture. *ACM SIGCOMM Computer Communication Review*, 40(2):26–33.

[Tyson et al., 2013] Tyson, G., Sastry, N., Cuevas, R., Rimac, I., and Mauthe, A. (2013). A survey of mobility in information-centric networks. *Communications of the ACM*, 56(12):90–98.

[Tyson et al., 2012] Tyson, G., Sastry, N., Rimac, I., Cuevas, R., and Mauthe, A. (2012). A survey of mobility in information-centric networks: Challenges and research directions. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 1–6. ACM.

[Wang et al., 2010] Wang, J., Peng, C., Li, C., Osterweil, E., Wakikawa, R., Cheng, P.-c., and Zhang, L. (2010). Implementing instant messaging using named data. In *Proceedings of the Sixth Asian Internet Engineering Conference*, pages 40–47. ACM.

[Wang et al., 2017] Wang, L., Tyson, G., Kangasharju, J., Crowcroft, J., Wang, L., Tyson, G., Kangasharju, J., and Crowcroft, J. (2017). Milking the cache cow with fairness in mind. *IEEE/ACM Trans. Netw.*, 25(5):2686–2700.

[Wouhaybi et al., 2013] Wouhaybi, R. H., Yarvis, M. D., Sharma, S., Muse, P., Wan, C.-Y., Prasad, S., Durham, L., Sahni, R., Norton, R., Curry, M., et al. (2013). Experiences with context management in emergency medicine. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(4):100.

[Xylomenos et al., 2014] Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., and Polyzos, G. C. (2014). A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049.

[Yu et al., 2013] Yu, Y.-T., Dilmaghani, R. B., Calo, S., Sanadidi, M., and Gerla, M. (2013). Interest propagation in named data manets. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1118–1122. IEEE.

[Yuan et al., 2012] Yuan, H., Song, T., and Crowley, P. (2012). Scalable ndn forwarding: Concepts, issues and principles. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE.

[Yürür et al., 2016] Yürür, Ö., Liu, C. H., Sheng, Z., Leung, V. C., Moreno, W., and Leung, K. K. (2016). Context-awareness for mobile sensing: A survey and future directions. *IEEE Communications Surveys & Tutorials*, 18(1):68–93.

[Zhang et al., 2013] Zhang, G., Li, Y., and Lin, T. (2013). Caching in information centric networking: A survey. *Computer Networks*, 57(16):3128–3141.

[Zhang et al., 2014] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B., et al. (2014). Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73.